A methodology for system integrity in design

A thesis submitted in partial fulfilment
of the requirements for the Degree of
Doctor of Philosophy in Mechanical Engineering
at the University of Canterbury,
Christchurch, New Zealand
by

Dirk Pons

26 September 2001

Dedication

To my wife

Tracey

and family

Arion, Ariel and Aiden,

for your love and support

without which this would have been impossible.

Abstract

The early stages of engineering design are formidable terrain for human designers as well as their methodologies and tools. The large uncertainties, fluidity of concept, and frequent lack of quantitative relationships and data make the formal design methods difficult to apply. This project sought to develop a methodology that could help manage the process.

The literature on the processes and methodologies engineering design were investigated, and interpreted with respect to a proposed generic design process. Existing methodologies are generally unsuitable for early design stages as they require relatively complete information and problem definition, which may be unavailable. There is need for methodologies that can support the designer even though the design uncertainties are high.

A methodology called *Design for System Integrity* (DSI) was developed to support the management of uncertainty at the early stages of design. Essential features of this methodology are the ability to simulate system behaviour (i.e. functional modelling) based on quantitative as well as qualitative information, the accommodation of uncertainty of analysis and process variability, and the provision of multiple viewpoints on the design, including but not limited to performance, cost, and reliability. The methodology is computationally demanding, and requires a supportive interface between the user and the algorithms, so it was necessary to embody it in software. The software development was from the ground up using the Delphi programming language rather than an expert system or database shell. The software development was a critical enabling mechanism in transforming the DSI methodology from a philosophy and design principle into an embodiment that could be explored and scrutinised.

The methodology was applied to a case study of dishwasher engineering design. One of the viewpoints in this model was wash performance, and the system was able

to successfully simulate this parameter in a probabilistic manner despite the uncertainties. Other views including electric shock hazard, reliability and cost were also simulated. Being able to produce not only an outcome but also measure its likelihood, is valuable information for evaluating and managing the integrity of a design. The DSI methodology may be applied in the early designs stages when uncertainty is high, as well as in the more mature stages.

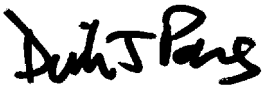
In addition a catalogue system was developed for functional modelling. This permits the designer to select a device from a database, and inherit the default properties of that device. Importantly, the functional modelling may occur over several viewpoints simultaneously, and placing a device into one view will cause the system to automatically create and populate additional viewpoints in the background with other properties of that device. It is also possible to substitute one device for another, corresponding to the testing of various candidates, and the properties of the device are changed to those of the new one.

The results demonstrate that it is possible to develop a methodology for modelling qualitative and quantitative performance at early design stages, so as to manage the integrity of a design.

Acknowledgements

Those who steer the boat are as crucial to the success of the voyage as those who work the engines. I could not have had better research supervisors than Professor John Raine and Associate Professor Kenneth Whybrew. The outcomes of this project are significantly due to their management skills, wisdom, balance, good nature, and broad mindedness. My sincere thanks to you both for your contribution to this work.

This research was supporting in part by the Public Good Science Fund (New Zealand) and Christchurch Polytechnic Institute of Technology, to whom grateful thanks are recorded.



Dirk Pons
Christchurch, New Zealand

First submitted: 24 February 2001
This Revision: 26 September 2001

Post script
Ken Whybrew died as this thesis was being submitted. He has been a great professional colleague and mentor on this project and his sudden passing has been felt keenly.

Contents

Chapter 1

Establishing the context 1
1.1 **Context** 2
1.2 **Models of the Design Process** 4
1.2.1 **Linear Model** 5
1.2.2 **Models of intra-organisational influence** 7
1.2.3 **Models of individual designer's activities** 11
1.2.4 **Phase diagrams** 12
1.2.5 **Project management** 12
1.2.6 **Extensions of project management: Design Structure Matrix and Signposting** 14
1.2.7 **Design Science** 17
1.3 **Comment on design models** 18
1.4 **Design mechanism and constraint diagram** 21
1.5 **Quality of information** 33
1.6 **Conclusions** 36

Chapter 2

Literature on design mechanisms and constraints 37
2.1 **Inventive mechanisms** 38
2.1.1 **Human based inventive processes** 39
2.1.2 **Functional decomposition** 39
2.1.3 **Mapping processes for inventiveness** 41
2.1.4 **Catalogue processes** 42
2.1.5 **Systematic idea generation** 43
2.1.6 **Morphological analysis** 46
2.1.7 **Concept invention with Artificial Intelligence** 47
2.1.7.1 **Logic based programming languages** 49
2.1.7.2 **Expert systems** 50
2.1.7.3 **Expert system case study: CAPE** 52
2.1.7.4 **Concept design using Grammars, Genetic Algorithms, and Simulated annealing** 54
2.1.7.5 **Concept design using experience, heuristics and case based reasoning** 58
2.1.7.6 **Neural networks** 59
2.1.7.7 **Artificial intelligence and uncertainty** 62
2.1.8 **TRIZ** 63
2.1.9 **Comments on inventive mechanisms** 67
2.1.9.1 **Distributed design** 68
2.1.9.2 **Need for quantitative information** 70
2.2 **Inventive constraints** 71
2.2.1 **Problem definition** 71

2.2.2 **Focus on the customer** . . . 73
2.2.3 **Quality function deployment** . . . 74
2.2.4 **Conjoint analysis** . . . 78
2.3 **Simulation and other Mechanisms to assess solutions** . . . 80
2.3.1 **Functional modelling** . . . 81
2.3.1.1 **Modelling energy, materials and signals** . . . 82
2.3.1.2 **Simulation of system behaviour** . . . 83
2.3.1.3 **Case study: Schemebuilder** . . . 88
2.3.1.4 **Feature based modelling** . . . 91
2.3.1.5 **Functional modelling and uncertainties at early design** . . . 94
2.3.2 **Assessing uncertainty and variability** . . . 94
2.3.3 **Fuzzy theory** . . . 96
2.3.4 **Decision Analysis and Belief Networks** . . . 98
2.3.5 **Constructing a Decision Analysis model** . . . 101
2.3.6 **System engineering, management science, quantitative analysis, and operations research** . . . 103
2.3.7 **Monte Carlo simulation** . . . 108
2.3.8 **Qualitative simulation** . . . 111
2.4 **Assessment constraints** . . . 112
2.4.1 **Quantitative assessment criteria** . . . 112
2.4.2 **Measurement scales** . . . 113
2.4.3 **Assessing qualitative outputs with bench marking** . . . 116
2.4.4 **Other qualitative assessment strategies** . . . 118
2.4.5 **Assessing multiple viewpoints** . . . 118
2.5 **Decision mechanisms** . . . 120
2.5.1 **Belief functions** . . . 120
2.5.2 **Classification of decision problems** . . . 122
2.5.3 **Conflict resolution** . . . 126
2.5.4 **Risk analysis** . . . 128
2.5.5 **Identifying and managing risk in the design process** . . . 130
2.6 **Recording and retrieving design intent** . . . 132
2.7 **Discussion** . . . 134
2.7.1 **Classifying design tools with the Generic Design Activity** . . . 134
2.7.2 **Characteristics of Early design** . . . 136
2.7.3 **Existing methodologies** . . . 137
2.7.4 **Uncertainty at early design** . . . 143
2.7.5 **Multiple viewpoints on design** . . . 146
2.8 **Conclusions** . . . 147

Chapter 3

Project strategy . . . 149
3.1 **Supporting early design** . . . 150
3.2 **Statement of hypotheses** . . . 151
3.3 **Solution approach** . . . 152
3.4 **Development milestones** . . . 154
3.4.1 **Initial approach** . . . 154

3.4.2 **DSI development** . . . 155
3.4.3 **Related developments** . . . 156
3.4.4 **Other features of DSI** . . . 157
3.5 **Application of the DSI methodology** . . . 158
3.6 **Conclusions** . . . 161

Chapter 4

Quantitative probabilistic computation . . . 163
4.1 **Introduction** . . . 164
4.1.1 **Interval analysis** . . . 164
4.1.2 **Single point probability variables** . . . 165
4.1.3 **Special methods for combining certain random variables** . . . 166
4.1.4 **Algebra of random variables** . . . 167
4.1.5 **Fuzzy theory** . . . 172
4.1.6 **Monte Carlo analysis** . . . 173
4.1.7 **Comment** . . . 174
4.2 **DSI probabilistic computation** . . . 177
4.3 **Implementation** . . . 181
4.3.1 **Software implementation** . . . 182
4.3.2 **User involvement** . . . 184
4.3.3 **Example results** . . . 187
4.3.4 **Interpreting results** . . . 189
4.4 **Validation against algebra of random variables** . . . 191
4.4.1 **Mathematical validation of DSI algorithms** . . . 191
4.4.2 **Robustness of DSI to low resolution simulation** . . . 194
4.5 **Comparison with Monte Carlo analysis** . . . 195
4.5.1 **Monte Carlo benchmark test** . . . 195
4.5.2 **Discussion of benchmark results** . . . 197
4.5.3 **Comparison with 'Analytica'** . . . 198
4.6 **Comparison with fuzzy theory** . . . 204
4.7 **Conclusions** . . . 209

Chapter 5

Qualitative probabilistic computation . . . 211
5.1 **Introduction** . . . 212
5.1.1 **Analysis uncertainty** . . . 212
5.1.2 **Process variability** . . . 213
5.1.3 **Exploring the wash performance issue** . . . 214
5.1.4 **Existing approaches to uncertainty of analysis** . . . 216
5.1.5 **Examples of decision analysis** . . . 219
5.2 **Probabilistic computation of maps** . . . 221
5.2.1 **Description of the method** . . . 221
5.2.2 **Implementation in software** . . . 226

5.2.3 **Applications for qualitative maps** 227
5.3 **Validation** 228
5.4 **Conclusions** 231

Chapter 6

Combining qualitative and quantitative probabilistic computation 233
6.1 **Introduction** 234
6.2 **Classifying quantitative and qualitative** 234
6.3 **Other approaches** 237
6.4 **Development strategy** 240
6.5 **Creating quantitative data from qualitative** 241
6.6 **Degrading the information content** 246
6.7 **Conclusions** 247

Chapter 7

Multiple viewpoints of design 249
7.1 **Introduction** 250
7.2 **Creating additional views as a by-product of functional modelling** . . 252
7.3 **Discussion on multiple views** 256
7.4 **Conclusions** 261

Chapter 8

Design for System Integrity Software 263
8.1 **Introduction** 264
8.2 **Representing the analysis problem in software** 264
8.2.1 **Creating a graph of the problem** 264
8.2.2 **Defining the input probability distributions** 266
8.2.3 **Propagating probability distributions** 267
8.3 **Using DSI for managing uncertainty** 268
8.3.1 **Interpreting the results** 269
8.3.2 **Extending to additional levels** 270
8.3.3 **Modelling qualitative relationships** 272
8.3.4 **Disseminating results** 273
8.4 **Conclusions** 274

Chapter 9

Software development 275
9.1 **Strategy for software development** 276
9.2 **Programme structure** 279

9.3 **Software milestones** 296
9.4 **Using Confidence intervals to express process variability** 298
9.4.1 **Defining subjective probability** 299
9.4.2 **Assessment of continuous probabilities** 300
9.4.3 **Use of confidence intervals for process variability** 301
9.4.4 **Implementation** 304
9.5 **Histogram Monte Carlo algorithm** 306
9.6 **Fuzzy arithmetic** 309
9.7 **Structure of 'prb' probability file** 309
9.8 **Validation of DSI software** 311
9.9 **Conclusions** 317

Chapter 10

Probabilistic approach to life cycle producer cost 319
10.1 **Introduction** 320
10.2 **Explanation of the DSI methodology** 322
10.3 **Model of Life Cycle costs** 323
10.4 **Results of Life Cycle cost analysis** 328
10.5 **Future work** 330
10.6 **Conclusions** 331

Chapter 11

Simulating Wash Performance 333
11.1 **Viewpoints of design** 334
11.2 **Dishwasher performance** 334
11.3 **ANSI/AHAM wash index** 335
11.4 **Development of a model for Wash performance** 336
11.4.1 **Strategy** 337
11.4.2 **Model development** 337
11.4.3 **Residual Soil patches - soil removal** 339
11.4.4 **Residual Soil particles - rinse effectiveness** 340
11.4.5 **Probabilistic computation of maps** 342
11.4.6 **Probabilistic computation of arithmetic operators** 345
11.5 **Calibration of wash model** 346
11.6 **Predicted wash performance of a different machine** 349
11.7 **Predicted Wash performance under other soil conditions** 351
11.8 **Discussion** 352
11.9 **Conclusions** 354

Chapter 12

Development of a Wash model 357
12.1 **Wash performance** 358
12.2 **Model details for Residual Soil patches - soil removal** 359
12.3 **Model details for Residual Soil particles - rinse effectiveness** 372
12.3.1 **Derivation of model for Dilution and Filtration of soil particles** 372
12.3.2 **Explanation of probabilistic model** 377
12.4 **Wash performance results** 382
12.5 **Conclusions** 384

Chapter 13

Simulation of dishwasher hydraulics at early design 385
13.1 **Introduction** 386
13.2 **Deterministic approach** 386
13.3 **Probabilistic approach** 389
13.4 **Results of probabilistic computation** 391
13.5 **Conclusions** 394

Chapter 14

Simulating dishwasher Reliability 397
14.1 **Introduction** 398
14.2 **Predicting Reliability at early design** 400
14.3 **Creating a reliability model as a by-product of functional modelling** . 403
14.4 **Conclusions** 409

Chapter 15

Warranty risk for dishwashers 411
15.1 **Existing approaches to warranty** 412
15.2 **Predicting Warranty exposure** 414
15.2.1 **Uncertain parameters for Weibull distribution** 414
15.2.2 **Warranty risk of domestic appliance** 416
15.2.3 **Uncertainty bands around Weibull parameters** 417
15.3 **Conclusions** 419

Chapter 16

Dishwasher hazards 421
16.1 **Dishwasher safety and hazards** 422
16.2 **Electric shock** 424
16.3 **Simulating shock hazard at early design** 432
16.4 **Fire hazard** 434
16.5 **Injury hazard** 437
16.6 **Hazard prevention** 441
16.7 **Product Liability** 445
16.7.1 **Cause-consequence model for product liability** 445
16.7.2 **Legal principles of product liability** 448
16.7.3 **Strategies against product liability** 452
16.8 **Conclusions** 454

Chapter 17

Discussion 455
17.1 **Identifying the need for assessment mechanisms** 456
17.2 **Evaluating the DSI methodology** 457
17.3 **Information quality for management of design** 460
17.4 **Summary of capabilities of the DSI methodology** 461
17.5 **Future research possibilities** 462
17.5.1 **Anticipating constraints in design** 462
17.5.2 **Other potential future research** 464
17.6 **Summary** 467

Chapter 18

Conclusions 469
18.1 **Project achievements** 470
18.2 **Closure** 471

Appendix 1

Operation of the DSI software 473
A1.1 **Introduction** 474
A1.2 **Selecting a distribution** 474
A1.3 **Multiple viewpoints** 477
A1.4 **Correlation** 478
A1.5 **Expert opinion** 480
A1.6 **Relationships supported in DSI** 480

A1.6.1 **Grid structure** 481
A1.6.2 **Probabilistic Operators** 482
A1.6.3 **Constant Operators** 484
A1.6.4 **Internal processes in Numerical analysis** 485
A1.6.5 **Textual (map) Operators** 488
A1.6.6 **Monte Carlo Operators** 489
A1.6.7 **Fuzzy theory Operators** 491
A1.7 **Genesis of distributions** 493
A1.7.1 **Genesis grid structure** 493
A1.7.2 **Generating the assert file** 494
A1.7.2.1 **Normal distribution** 494
A1.7.2.2 **Weibull distribution** 498
A1.7.2.3 **Beta distribution** 503
A1.7.2.4 **Triangle distribution** 505
A1.7.2.5 **Uniform distribution** 505
A1.8 **Discussion** 506

Appendix 2

Test data 507
A2.1 **Chi square test on validation example** 508
A2.2 **Chi square test on robustness example** 511
A2.3 **Scoring wash tests in the DSI model** 512
A2.4 **Chi square test for goodness of fit wash percent** 516

Appendix 3

Published papers 519
A3.1 **IMechE paper** 520
A3.2 **ICED paper** 532

References 543

Index 566

List of figures

Figure 1.1: Linear model of design 5
Figure 1.2: Hales' model of the design process. From Hales (1994). 8
Figure 1.3: Raine's model of design activity. This model explores the integration of marketing and manufacturing with design activities. From Raine (1998). 9
Figure 1.4: Model of innovation in design, from Crisp (1986) 10
Figure 1.5: Spiral design process, according to Candy et al (1996) 11
Figure 1.6: Phase diagram, modified from Raine et al (1998). 12
Figure 1.7: Gantt chart showing project plan 13
Figure 1.8: Model to illustrate DSM 14
Figure 1.9: Design mechanism and constraint diagram at the top level. 23
Figure 1.10: Design mechanism and constraint diagram for the activity of 'Develop product'. 26
Figure 1.11: Generic Design Activity in the engineering design context. 29
Figure 1.12: Model of the decision process, in particular the influences on a designer or design manager who is selecting a solution. 34
Figure 2.1: Inventive mechanisms 38
Figure 2.2: Functional decomposition process 41
Figure 2.3: Multi-layered logic, from Clibbon et al (1996) 42
Figure 2.4: Systematic idea generation process. 44
Figure 2.5: Problem definition 72
Figure 2.6: Quality Function Deployment uses a matrix to show the relationships between customer requirements (rows) and engineering characteristics (columns). The triangular matrix at the top is used to correlate engineering characteristics with each other. Other parts of the matrix show analysis of competitors products, and target values for the product. From Roland Andersson in Bergman & Klefsjö (1994). 75
Figure 2.7: QFD aims to progressively develop the product from customer requirements through to Production requirements. The output from one matrix is fed in as the input to the mext. From Bergman & Klefsjö (1994). 76
Figure 2.8: Nodal view from Zhong and Dooner (1996). 86
Figure 2.9: Bond graph model of DC servomotor at its second level of complexity, using Simulink, Bracewell at al (1989b). 89
Figure 2.10: Function structure from Brady & Juster (1996) 93
Figure 2.11: Bayesian belief network created with 'Hugin' to model pump faults. 99
Figure 2.12: An Influence diagram shows decision nodes (i.e. variables that the user has control over, as rectangles), chance events that are uncontrolled (ellipses) and utilities (i.e. outcomes, as ovals). This diagram courtesy of Analytica (http://www.lumina.com). 99
Figure 2.13: A decision tree for modelling market value of a product. The tree is read from left to right, and each split represents a decision or a chance external event. This diagram courtesy of Analytica (http://www.lumina.com). 100
Figure 2.14: Decision analysis is applied to the failure of a pump, with the decision graph shown at right. In this example it has been asserted that the pump is leaking ('Water Leaks' = 1, at bottom left), and after propagation the likely causes of the fault are displayed in the list at left, along with their probabilities. 102
Figure 2.15: Analytica is a decision analysis tool that is used to model risk events. This tutorial example models a purchasing decision: whether to buy a product that can be upgraded, as opposed to the basic product. The model is shown at top left, as a graph connecting various decision and chance events. The user provides the relationships that describe the graph, and the probabilities of various outcomes, and the system calculates the final result. The result in this case is present value, and this is shown by the chart at bottom right (cumulative probability). The details of one decision event are shown at top right and bottom right. 110
Figure 2.16: Checklist for bench marking. Attributed to Watson in Bergman & Klefsjö (1994). 117
Figure 2.17: Raine's model of recursive exploration of alternatives. This model specifically addresses the issue of checking the design from multiple viewpoints, and is a development of Pugh's Total Design activity model (Pugh, 1991) with concurrent design, manufacturing planning and market development, plus increased focus on customer needs and the market. Adapted from

Raine et al (2001) with some detail omitted for clarity. 119
Figure 2.18: Taxonomy of design, adapted from Ullman and D'Ambrosio (1995), showing possible attributes of a design problem from the perspective of decision making. 124
Figure 2:19: Generic Design Activity showing the positions of various tools and support systems in the design process. 135
Figure 4.1 Input distribution A. 178
Figure 4.2 Input distribution B. 178
Figure 4.3 Output distribution C. 180
Figure 4.4: Cumulative probability for output. 181
Figure 4.5: Flowchart for the DSI method. 186
Figure 4.6: Summation of two input probability densities. 187
Figure 4.7: Subtraction of two input probability densities. 187
Figure 4.8: Multiplication of two input probability densities. 188
Figure 4.9: Division of two input probability densities. 188
Figure 4.10: Lesser of two input probability densities. 188
Figure 4.11: Greater of two input probability densities. 189
Figure 4.12: Probability densities for 'lesser' operator 190
Figure 4.13: Cumulative probability for 'lesser' outcome. 190
Figure 4.14. Probability distribution resulting from the quotient of two identical uniform distributions (0,1), as determined by the algebra of random variables. 192
Figure 4.15. Probability distribution resulting from the quotient of two identical uniform distributions (0,1), as determined by DSI. 193
Figure 4.16. Probability distribution resulting from the quotient of two identical uniform distributions (0,1), as determined by DSI, with only 10 data points. 194
Figure 4.17: Results from DSI numerical benchmark with smoothing (proportional sort). Computation time was 4.340 s. 196
Figure 4.18: Results from DSI numerical benchmark, with no smoothing. Computation time was 1.600 s. 196
Figure 4.19: Results from Monte Carlo algorithm using 10000 output bins. Computation time 5.220 s. 197
Figure 4.20: Results from Monte Carlo algorithm using 1 000 000 output bins. Computation time 2 min 33.070 s. 197
Figure 4.21: Reliability model as it appears in Analytica. 198
Figure 4.22: DSI view of reliability of two bulbs (in electrical parallel), in series with a switch. 199
Figure 4.23: Reliability model as it appears in Analytica. 199
Figure 4.24: System reliability by DSI, mean 15 hrs, and median 12 hrs. 200
Figure 4.25: Results from Analytica simulation for system life. 200
Figure 4.27: Analytica results for a fragment of the 'Seat belt safety.ana' problem. 201
Figure 4.26: DSI results for a fragment of the Seat belt safety sample problem. 201
Figure 4.28: A set of cumulative probabilities may be used in Analytica to define a distribution. 202
Figure 4.29: The DSI equivalent probability density distribution. 203
Figure 4.30: DSI result for summation of two triangular probability density distributions. 204
Figure 4.31: Addition of two variable parameters using Fuzzy theory. 205
Figure 4.32: Comparison of results for a probabilistic computation and fuzzy theory. 207
Figure 4.33: Comparison of results for a product operation using probabilistic computation and fuzzy theory. 208
Figure 5.1: Decision analysis applied with Hugin. 219
Figure 5.2: Probability table from a sample file in Analytica. The 'Technical Success Uncertainty' is the output and it may be 'Success' or 'Failure'. Only the 'Success' slice of the table is shown here. The 'R&D Strategy' may be 'hi tech', 'mixed tech' or 'low tech'. The 'Technology' may be 'Ceramic Turbine' etc. The table gives the probability of 'Success' given inputs for 'R&D Strategy' and 'Technology'. For example, 'hi tech' and 'Ceramic Turbine' have a 0.29 probability of success (and a 0.71 probability of failure, not shown). 220
Figure 5.3: View of map operator in Design for System Integrity (DSI) software, showing that two inputs, Soil type and Soil thermal treatment are used to determine Soil state. 221
Figure 5.4: Soil type shown as a histogram based on mass of various soils used in the ANSI/AHAM test. 222
Figure 5.5: The profile for Soil thermal treatment can range from 'fresh' to 'burned' soil. For the

ANSI/AHAM test there is only a single value: 'dried'. In this particular example it has also been asserted that some of the soil is 'fresh' and some 'reheated'. 222
Figure 5.6: Map to convert soil type and soil thermal treatment into soil state. Numbers are probabilities where any one column will sum to unity. 224
Figure 5.7: Soil state is a combination of soil type and soil thermal treatment. It describes the important wash characteristics of the soil. 225
Figure 5.8: Graph for determining Weather from Season and Wind. 228
Figure 5.9: Asserted probability for qualitative parameter Season. 229
Figure 5.10: Asserted probability for qualitative parameter Wind. 229
Figure 5.11: Calculated probability for Weather. 230
Figure 6.1: Classification system used in this thesis to distinguish between qualitative and quantitative variables. Quantitative variables present no special difficulty as they are those for which arithmetic operators are valid. The qualitative variables are ordinal or nominal, and simultaneously either numerical or textual. Examples of each type are given in brackets. The arrows indicate the direction of decreasing information content. 236
Figure 6.2: Model used to determine soil removal time (A) using a map. The graph in the centre of the figure shows the operation. The inputs are the two distributions at the bottom of the figure, and the output is the top distribution. 242
Figure 6.3: Fragment of Map used to correlate two input variables, soil state and wash temperature, to soil removal time. 243
Figure 6.4: Bridging from quantitative to qualitative variables. The graph model (upper left) uses a map to combine two nominally qualitative variables. In this case one of the variables (Dishware.Daylight) at lower left is quantitative since it is the result of a previous computation (not shown). The map expects only a few values of Dishware.Daylight, not as many as present in the quantitative input. However the software copes with this by distributing the quantitative input into the coarser quantitative bins and then proceeding with the map computation. The other input (lower right) is purely qualitative. The final output at top right is qualitative (viz the histogram bars) but the map has given its text labels a numerical interpretation so it is eligible to be input to subsequent quantitative operations. 247
Figure 7.1: Multiple views created by the DSI are shown here. The original view was 'Viewpoint' and a device 'PumpACME' was placed on it. Since the type of the device was set to 'WashPump' which has data for multiple views in the domain file, the system automatically created all the other views (windows) and placed the device there too. There is default probability data behind each device. 254
Figure 7.2: On placing a device (eg 'PumpACME') on a view, the system prompts for the type. In this case the user selects 'WashPump' and will therefore get all the views and data that belong to that generic type. 255
Figure 8.1: The Graph tab sheet is the primary interface for the user. On its canvas the user draws the graph representing the problem. The graph here shows that Profit = Income - Cost. This is created using the buttons on the toolbar. 265
Figure 8.2: The Genesis tab sheet is used to define the inputs in terms of probability distributions. For example the grid here shows that Income is a Normal distribution with mean 1000, standard deviation 30 and that it will be modelled as far out as 3 sigmas (standard deviations) on each side of the mean. 266
Figure 8.3: Propagation involves taking the input probability distributions and combining them using the given relationship (minus in this case) using the DSI probabilistic algorithm, to produce the output distribution. The block diagram graph at left shows the computation group, and the chart at right shows the resulting probability distribution for Profit. 268
Figure 8.4: This chart shows the density and cumulative distributions for 'Profit'. The density is the lighter curve. The density has been artificially enlarged in height to make it clearer. 270
Figure 8.5: Deeper levels of detail may be created as the model is further developed. This graph extends the previous example by adding sub calculations to replace the parameters that formerly were assumed to be given distributions. Note the use of additional mathematical operators such as product (*) and plus (+). The model may be extended indefinitely. 271
Figure 8.6: Linked trees may be set up in DSI. In this case the top level graph (left) contains the entry 'Cost.Overheads' and this is a separate sub tree which shows additional detail (right). There is only one probability file for 'Cost.Overheads', and it is accessed by both graphs as needed. 272

Figure 8.7: DSI is able to automatically create web based image maps such as this. These maps may be viewed with any browser. A mouse click on a blocks loads an image of the probability distribution for that parameter (or a sub tree if appropriate). 274
Figure 9.1: Overview of DSI method. 281
Figure 9.2: Details of create graph event 283
Figure 9.3: Details of mouse down event 285
Figure 9.4: Details of how new relationship panel is drawn 286
Figure 9.5: Details of how new genesis panel is drawn 287
Figure 9.6: Details of mouse down event on a panel 288
Figure 9.7: Details of left click event on a panel 289
Figure 9.8: Details of simulate system event 291
Figure 9.9: Details of qualitative calculation 293
Figure 9.10: Details of quantitative calculation 295
Figure 9.11: Confidence interval fitted by Weibull distribution. Oval markers show measures and probabilities provided by user, and thick line shows fitted curve (cumulative distribution). Thin line shows density function (not to scale) for comparative purposes. 302
Figure 9.12: Weibull fit to different parameters. 304
Figure 9.13: Weibull estimation software 305
Figure 10.1: Top level model for producer profit is nett total profit less venture costs. 324
Figure 10.2: Breakdown of Venture costs into hierarchical cost elements. Computation begins at the bottom of the tree and progresses upwards. Confidence limits are given for all the primary assertions, and these are shown in boxes. 325
Figure 10.3: Venture cost as a probability density distribution, produced by propagating the assertions though the model shown in the previous figure. 325
Figure 10.4: Nett total profit is computed as a sequence of operations on asserted inputs. This tree shows the computational structure, and the confidence estimates used for the assertions. 326
Figure 10.5: The cost of a dishwasher in terms of the component subsystems is modelled by this graph. 327
Figure 10.6: Probability density distribution for Nett total profit after simulation. 327
Figure 10.7: Producer profit is the nett total profit less the venture costs. This figure illustrates the probability distributions involved in the inputs, and the resulting probability distribution. . . . 328
Figure 10.8: Producer profit, shown as a cumulative probability distribution. 329
Figure 10.9: Possible model of Consumer's costs 330
Figure 11.1: Wash performance graph. 338
Figure 11.2: Graph for number of demerit points for patches of soil. 339
Figure 11.3: Graph for demerit points for particles. 341
Figure 11.4: Representation of a map relationship in the DSI software. 342
Figure 11.5: Soil type shown as a histogram based on mass of various soils used in the ANSI/AHAM test. 342
Figure 11.6: The profile for Soil thermal treatment can range from 'fresh' to 'burned' soil. For the ANSI/AHAM test there is only a single value: 'dried'. 343
Figure 11.7: Map to convert soil type and soil thermal treatment into soil state. 344
Figure 11.8: Result for Soil state, as a combination of soil type and soil thermal treatment. 344
Figure 11.9: Probabilistic addition in DSI. 345
Figure 11.10: Result (heavy line) after addition of two input probability distributions (light lines). . . . 345
Figure 11.11: Interior view of ASKO 1805 dishwasher. 346
Figure 11.12: Actual wash performance for the ASKO 1805. 347
Figure 11.13: DSI predicted wash performance for the ASKO 1805. 348
Figure 11.14: Wash performance for the ASKO 1805 with the bars showing measured data, and the points showing the DSI simulation results. 348
Figure 11.15: Simulated wash performance for the GE machine. 349
Figure 11.16: Wash performance for the GE Profile with the bars showing measured data, and the points showing the DSI simulation results. 350
Figure 11.17: Modified food profile to simulate the profile that a typical family might generate. 351
Figure 11.18: Simulated wash performance for the ASKO 1805 operating on the modified soil load. 352
Figure 12.1: Wash performance model. 358

Figure 12.2: Graph representing demerit points for patches of soil. 359
Figure 12.3: Soil pre-treatment with no rinse. 360
Figure 12.4: Wash temperature profile. 361
Figure 12.5: Soil removal time A. 362
Figure 12.6: Detergent concentration. 363
Figure 12.7: Jet velocity, in m/s. 363
Figure 12.8: Soil removal time after converting to a numerical scale. 363
Figure 12.9: Wash time. 364
Figure 12.10: Soil residual fraction. 365
Figure 12.11: Sprayer main type is a moving jet for most dishwashers. 366
Figure 12.12: Wash coverage is determined from the three sprayer systems. 366
Figure 12.13: Dishware daylight. 367
Figure 12.14: Wash direct fraction. 367
Figure 12.15: Wash cavity volume in cubic metres. 368
Figure 12.16: Wash power density [W.m^{-3}] 369
Figure 12.17: Dishware wash coverage N as a fraction. 369
Figure 12.18: Soil wash coverage as a fraction. 370
Figure 12.19: Soil patch residual fraction as a fraction. 370
Figure 12.20: Soil patch input. 371
Figure 12.21: Soil patch weight. 371
Figure 12.22: Demerit points attracted by a dishware item. 372
Figure 12.23: Computation graph for the removal of soil particles by rinse and filtration. 376
Figure 12.24: Filter ratio for the model. 377
Figure 12.25: Total Water retained volume. 377
Figure 12.26: Water fill volume (per fill). 378
Figure 12.27: Water retained original fraction. 379
Figure 12.28: Soil redeposited fraction. 379
Figure 12.29: Soil particles initial. 380
Figure 12.30: Total demerit points for particles on an item of dishware. 381
Figure 12.31: Total soil demerit points for particles on an item of dishware. 382
Figure 12.32: Discrete addition of two input curves (light) to produce an output (thick curve). 383
Figure 12.33: Wash performance [%] predicted for this set of model parameters. 383
Figure 13.1: Graph of relationships used to solve the probabilistic reasoning aspects of dishwasher hydraulic performance. The graph shows the computation steps necessary to determine the top parameters. Shaded blocks show parameters that are primary assertions, that is information that the user has to provide. Some sample data are shown as estimate pairs (value, cumulative probability). Unshaded blocks are intermediate values that result from calculations on the assertions and the constants. In some cases an intermediate parameter has a specific interpretation, in which case it is labelled as such, eg area is determined from pipe diameter and some constants. However many other intermediate values have no significant physical interpretation, and are labelled starting with 'X'. 390
Figure 13.2: Final result for Nozzle Jet velocity, after probabilistic computation. This is a probability density, which should be interpreted as a histogram. The horizontal axis is jet velocity [m/s] and the vertical axis is probability density. The shape of the distribution shows a peak at 2.6 m/s which is the mode. There is no lower tail to the distribution below 0 m/s, and this is consistent with the physical interpretation that jet velocity may not be negative. The distribution does however have a long upper tail, so that velocities in the region of 20 m/s are possible, but with low probability. Neither the mean nor the median are apparent from inspection of this chart. The value of this chart is that it shows the shape of the distribution and where the results are concentrated. 392
Figure 13.3: Cumulative probability for Nozzle Jet velocity. This is a companion chart to the density shown above. The horizontal axis is jet velocity [m/s] and the vertical axis is the cumulative probability (area under the density curve up to and on the left of the point of interest). Cumulative distributions usually have an "S" shape, which this one displays to some extent. The cumulative distribution allows risk to be assessed. For example the probability of a velocity less than 2 m/s may be determined by inspection as 17%. Likewise the probability of a velocity greater than 20 m/s may be read off the chart as (1-0.97)=0.03 = 3%. Another important statistic is the median, which is the velocity for 50% probability, and this is 4.6 m/s by

inspection. Half the time the velocity could be expected to be below this value, and half the time above it. Other percentiles may also be determined from the chart. However the mean is not apparent from inspection of this chart. The value of this chart is that it shows the risk of various outcomes, and permits those risks to be quantified. 394
Figure 14.1: The DSI software will automatically create part of a reliability model (lower view) when relevant devices are used in another viewpoint (upper view). 403
Figure 14.2: The designer can connect up the provided boxes with suitable relationships to determine system reliability. 404
Figure 14.3: Graph using 'min' ('or') function. 405
Figure 14.4: Result (heavy line) after combining two distributions (light lines) using the 'min' function. 406
Figure 14.5: Failure probability of whole dishwasher, predicted on the basis of generic data for constituent device failure distributions. 407
Figure 14.6: Cumulative failure distribution for generic dishwasher. Time in years is on the horizontal axis and cumulative probability on the vertical. 408
Figure 15.1: Weibull cumulative distribution fitted to experimental data. The horizontal axis is load cycles and the vertical is probability. The markers are data points and the heavy curve is the fitted Weibull curve. 416
Figure 15.2: Model of warranty exposure with uncertain Weibull parameters. The function 'warranty' computes the Weibull equation using the uncertain input variables. 418
Figure 15.3: Model of warranty exposure with uncertain parameters in Weibull equation. The horizontal axis is the reliability at 900 cycles and the vertical is probability as a histogram. 418
Figure 16.1. The main hazards for consumer appliances are electric shock, fire, and injury, as this fault tree shows. 422
Figure 16.2. Electric shock requires that two events occur, Access to the hazard, and that the Part be electrically energised. 424
Figure 16.3. Access to the hazard is one half of the electric shock hazard. This fault tree shows various potential causes whereby a person can gain access to an electrical hazard. 425
Figure 16.4. Electrically energised parts are necessary for electric shock to occur. This figure shows contributory causes. 426
Figure 16.5: Leakage current requires that an electrical leak source exists, and also that the earthing on the part be inadequate. Each of these have lower level events. 427
Figure 16.6: Inadequate earthing has a number of causes, any one of which can cause the fault. 428
Figure 16.7: Fault tree for Insulation failure 429
Figure 16.8: Wetting of a domestic appliance is a real possibility that needs to be designed for. Various foreseeable fault conditions are shown here. 429
Figure 16.9: Insulation damage involves several possible causes, of which damage due to mechanical cause needs to be an important design consideration. 430
Figure 16.10: Stray current can be caused by one of several fault conditions. 431
Figure 16.11: Model of electric shock hazard, as expressed in DSI. Default data (purely representative) were provided for each of the prime events. 433
Figure 16.12: Results of probabilistic computation of electric shock hazard, giving time to this event. The horizontal axis is time [years] and the vertical axis is histogram probability. The model predicts a mean time of about 11 years for an electric shock to occur, but it should be remembered that the data used here were only representative. 434
Figure 16.13: Fire fault tree 435
Figure 16.14: Fault tree for causes of a part becoming excessively hot 436
Figure 16.15: Fault tree for injury 437
Figure 16.16: Fault tree for usage 439
Figure 16.17: Fault tree for the existence of potential hazards. 440
Figure 16.18: Fault tree for hazard prevention failure. 441
Figure 16.19: Fault tree for failure of hazard prevention systems 442
Figure 17:20: Fault tree for safety system disabled. 442
Figure 16.21: Fault tree for warning system failure. 443
Figure 16.22: Consequence diagram for product liability. 445
Figure 16.23: Consequence diagram for the process of product liability. 446
Figure 16.24: Legal principles of product liability represented as a fault tree. 449

Figure A1.1: Correlation of pump flow to pump pressure using a map. 479
Figure A1.2: Map for determining pump flow (1, 3, 5, 7) from pump pressure (25, 75, 125, 175) for a particular value of pump power. 479
Figure A1.3: Relationship grid. 481
Figure A1.4: Example of Monte Carlo simulation. 490
Figure A1.5: Model and result using Fuzzy theory. 492
Figure A1.6: Genesis grid. 494
Figure A1.7: The normal distribution. 495
Figure A1.8: Weibull density with characteristic life 100 and shape factor 3. 499
Figure A1.9: Exponential distribution obtained with Weibull characteristic life 100 and shape factor 1. 501
Figure A1.10: Beta distribution (8,5) giving a bell shaped curve. 503
Figure A1.11: Triangular distribution (3,4,9) 505

List of tables

Table 1.1: Design Structure Matrix for a fragment of the model shown in the previous figure. 15
Table 2.1: Classification of decision problems, adapted from Ullman and D'Ambrosio (1995). 123
Table 4.1: Outcomes for all combinations of inputs A and B. 179
Table 4.2: Benchmark test using subtraction of two Weibull distributions. 196
Table 4.3: DSI data file. 203
Table 5.1 Map for determining Weather from Season and Wind. 228
Table 9.1: Extended Pearson-Tukey method 300
Table 9.2: User estimates 302
Table 9.3: Weibull transformations 303
Table 9.4: New estimates 304
Table 11.1: ANSI/AHAM demerit points depend on the particle size. 335
Table 15.1: Partial data set, with hazard analysis equation, for determining Weibull parameters. . . 416
Table A1.1: Two and three dimensional operators supported by DSI. 483
Table A1.2: The constant operators supported by DSI. 485
Table A1.3: Monte Carlo operators supported by DSI. 491
Table A1.4: Fuzzy operators supported by DSI. 493
Table A1.5: Grid parameters for Normal distribution. 496
Table A1.6: Interpretation of Weibull parameters. 502
Table A2.1: Data for DSI and the algebra of random variables. 510
Table A2.2: Data for DSI with ten data points, and the algebra of random variables. 511
Table A2.3: Raw wash data scores for ASKO 1805, Detergent Amounts 15 & 30g, Programme Normal, test reference CP:11 of 15/07/1998. The numbers are the demerit points for each of up to 10 dishware items. The worst possible score in this test is 9. 512
Table A2.4: Count of various fault (0-9) occurrence, for different categories of ware. 513
Table A2.5: Frequency of fault is used to determine the wash index for each ware type, and then the overall wash index. 513
Table A2.6: Data for two wash tests is shown here, with only the scores for crockery included. Scores have been stretched to a 0 to 10 scale on a proportional basis. 514
Table A2.7: Count of various fault scores, based on the data in the previous table. 515
*Table A2.8: Data for wash performance histogram. Wash percent is determined from the demerit point score as (10-score)*10. Frequency is determined from the Count column of the previous table, divided by the total number of faults. 515*
Table A2.9: Data for simulated wash performance. The first column is the wash performance [%] and the second column is the frequency. This data is the tabular version of the graph shown previously in Figure 11.13. 517
Table A2.10: Calculation of Chi square for simulated wash performance. 517
Table A2.11: Calculation of test statistic. 518

Nomenclature

x	monetary value at year n	114
r_i	interest- or discount-rate	114
n	number of years to when value x will be received	114
x_i	income at year i (negative values may be used for expenses)	115
R(x,y)	region in the x-y plane	167
R(t)	Soil residual fraction	364
D1	input distribution one	365
D2	input distribution two	365
p	fraction (percentile)	365
S_o	initial number of particles (i.e. soil load)	373
V_f	volume of water per fill	373
V_d	drain volume	373
N_e	number of wash cycles plus number of rinse cycles	373
S_{ew}	number of soil particles on dishware	373
V_w	volume of fluid as film on dishware	373
F_f	filtration ratio (the fraction of fluid that is filtered)	374
n_f	filtration efficiency (the smallest particle removed)	374
R_f	reliability of filtration at a given time	374
Q	discharge	387
A_{nozzle}	area of one nozzle	387
D_{nozzle}	diameter of nozzle	387
N_{nozzle}	number of nozzles	387
C_c	nozzle coefficient of contraction	387
H_{mano}	manometric head	387
g	acceleration due to gravity	387
w	rotation speed	387
$D_{impeller}$	diameter of wash pump impeller	387
H_{lift}	static lift height from wash pump to nozzles	388
k_{bend}	shock factor for plumbing system, assuming that this includes shock and friction	388
N_{bend}	number of bends in plumbing system	388
k_{nozzle}	shock factor for nozzle	388
C_v	nozzle velocity coefficient	388
R(t)	reliability, that is probability of survival at operating cycle t	414
F(t)	unreliability, that is probability of failure at operating cycle t	414
π = 3.14159		496
e = 2.7183		496
x	variable	496
u	mean	496
s	standard deviation	496
α	constant	499
β	shape factor in Weibull equation	499
η	characteristic life in Weibull equation	499
λ	failure rate	501

Chapter 1

Establishing the context

This chapter describes the design process and the context in which this project seeks to assist the process. Various models of the design process are discussed, including the linear model, organisational process models, phase diagrams, project management models, and design structure matrix. Another model termed a design mechanism and constraint diagram is developed, and one of its internal components called the Generic Design Activity is detailed to provide a framework for systematically grouping the design tools.

1.1 **Context**

The context of this thesis is the early stages of engineering design, and particularly the mechanisms (methodologies and tools) that assist the process. Early design refers to the development of a concept design as opposed to a detailed design. The early design stages are characterised by large uncertainties, and many challenges are presented to the deployment of formal design mechanisms.

There is a need for a methodology that would assist the design manager to assess the integrity of an engineering system at the early design, and from multiple viewpoints. It is especially important to be able to accommodate uncertainties in this process: uncertainties may present as *process variability* or as *uncertainty of analysis*. The latter refers to a lack of quantitative relationships and this characteristic of early design perhaps more than any other is particularly problematic for existing design methodologies.

Available tools to assess the integrity of a design include the various *risk assessment* mechanisms. These include failure mode and effect analysis (*FMEA*), fault tree analysis (*FTA*), *decision analysis*, and various statistical tools for *reliability* analysis. However these are not so much tools to assist in the development of a design as tools to assess well-developed existing designs. Thornton et al (2000) surveyed United States industry practices on variation risk management and discovered that the practices were typically only applied late in the design process. They identified the need for better quantitative methods to assess product risk. The methods need to be easier to use and populating data (eg process capability and its uncertainty) need to be available. They also identified the need for methods to help designers ‘develop a system view of variation ... capable of handling the complex *flowdowns* that typify manufactured products’ (p141). Flowdown is a systematic *decomposition* process to identify subsystem (and sub process) *key characteristics* (which may be design parameters or features or risk factors) that contribute to the overall product performance.

Other tools used during design include quality function deployment (QFD), analytical hierarchy process (AHP), artificial intelligence (AI), decision analysis, fuzzy theory, and Monte Carlo analysis to name a few. These and other aspects of design methodologies and their tools will be further discussed in this thesis, and work on a new methodology called Design for System Integrity (DSI) is described.

The balance of this chapter reviews existing design processes and their usefulness at early design. It seeks to explore the issue whether the design process is sufficiently well understood to be able to augment or even automate part of it. This is followed by discussion on desirable but not yet existing mechanisms: if there were to be tools to help manage the engineering design process, what would we expect those tools to do? A Generic Design Activity is then proposed, and used as a framework for classifying the existing design methodologies. The discussion continues in chapter two with an investigation of design tools and where they are positioned on this framework. In this way the capabilities and limitations of the existing design tools are identified, and the required characteristics of new tools identified. This leads in chapter three to a statement of hypothesis regarding a proposed new methodology called Design for System Integrity (DSI), with a description of the solution approach. The way in which this methodology accommodates both quantitative and qualitative[1] relationships and data is then described in chapters four to six. The ability of the methodology to simultaneously generate and populate multiple viewpoints is described in chapter seven. The methodology is computationally demanding, and to explore and demonstrate its feasibility it was necessary to embody it in software. The operation of this software from the user's perspective is described in chapters eight (with further details in appendix 1). The software development is described in chapter nine. The software development was a large undertaking and a critical enabling mechanism in transforming the DSI methodology from a philosophy and design principle into an embodiment that can be explored and scrutinised.

[1]The interested reader is referred to Chapter 6 for a description of how the terms quantitative and qualitative are used in this work.

There follow application chapters, ten to sixteen, in which the methodology is demonstrated. Producer cost (chapter ten), wash performance (chapters eleven, twelve and thirteen), reliability (chapter fourteen and fifteen) and hazards (chapter sixteen) are discussed. If any one chapter was the crux of this thesis it would be chapter eleven on wash performance, since it demonstrates the capabilities of the methodology as regards combining quantitative and qualitative relationships. Also, it presents a calibrated model. The thesis closes with a discussion (chapter seventeen) primarily on managing the design process, and then brief conclusions (chapter eighteen). At the end an index has been provided to assist the reader navigate the thesis, since a study of design methodology inevitably transects knowledge domains.

1.2 **Models of the Design Process**

The process of *engineering design*, namely the creation of instructions for the fabrication of machine systems that provide utility, remains an enigmatic aspect of human intelligence despite many studies[2]. There are several schools of thought regarding the *design process*. Some view design as chaotic, creative, and intuitive, and likened to the artistic process. Under this view design needs to be unshackled from constraints, and is intractable as regards being managed. Others view design as systematic and *discursive*[3], and as such a process which is fundamentally

---

[2]Design is a creative activity with many facets, as evident in the multitude of existing definitions of design. The author has avoided providing a detailed definition of design, in the expectation that most readers will be familiar enough with the concept. However if pushed for a definition, the author would adopt that of Hubka and Eder (1996) who wrote: '*The task of designing consists of thinking ahead and describing a structure, which appears as (potential) carrier of the desired characteristics (properties, particularly the functions). One can express this statement also in process terms: designing is defined as the transformation of information from the condition of needs, demands, requirements and constraints (including the demanded functions) into the description of a structure which is capable of fulfilling these demands. The demands must include the wishes of the customers, but also all stages and requirements of the life cycle and all intermediate states that the product must pass through*' (p4). The reason for preferring this definition set is that it includes the concept of structure being a carrier for function, it acknowledges the origins of the demands (including those of the customer), it pays more attention to constraints than many other definitions of design, and it incorporates the life cycle considerations (viz different viewpoints). Hubka and Eder (1996) have also compiled an extensive list of other definitions of design.

[3]The disciplined application of methods rather than intuition.

manageable. Yet others feel the process should be free of any imposed process, and thus there exists some conflict and fragmentation in the design community (Finger and Dixon, 1989 a). However the viewpoints are not necessarily contradictory (Pahl & Beitz, 1988) and there could be a combination of creativity and method in all design activities. There exist various models[4] of the design process, principal features of which are described next.

1.2.1 **Linear Model**

The engineering design process starts with product definition and passes through concept design, into detailed design. *Concept design* is generally understood to be the manipulation of ideas and principles that are relatively abstract and ill-defined with the intent of finding and selecting a solution principle (sometime called Early Concept design), followed by the development of broad solution schemes wherein means have been established for performing the major functions (*embodiment design*, or *layout design*). Expressed differently, concept design transforms of functional requirements into a nonspecific configuration (without dimensions or other specific attributes) or physical embodiment.

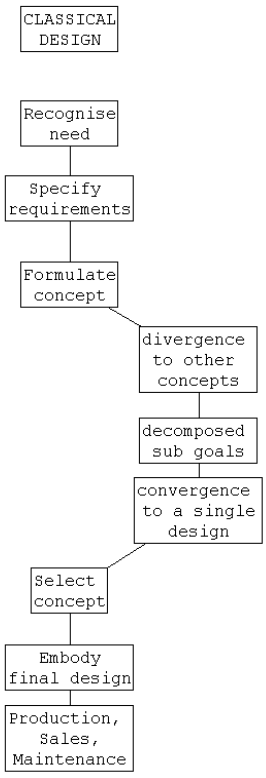


Figure 1.1: Linear model of design

Detailed design involves firming up details in the concept, perhaps initially with no particular values assigned to the principal attributes (*configuration design*, or *structure design*), but eventually the assignment of specific values to attributes

[4]The models are intended to help manage, anticipate and plan the process. The models are generally taken as guidelines rather than prescriptive methods, although this varies between design cultures and organisations.

(*parametric design*). The output of the Detailed design process is a set of instructions that are sufficient for the fabrication and assembly of the desired system. Quite where concept design leaves off and detailed design takes over is subject to different interpretations in the literature, and it is probably prudent to acknowledge that there is a spectrum of activities which may be partitioned in various ways.

The *linear model* (or systematic model) of design sees the process as comprising activities that are undertaken in sequence, as illustrated in Figure 1.1. For example Finger and Dixon (1989 a, b) describe the design process as:

- recognise need,
- specify requirements,
- formulate concept, consisting of sub-tasks of divergence to other concepts, transformation into decomposed sub goals, and convergence to a single design.
- select concept,
- embody final design,
- production, sales, maintenance

The viewpoint taken by the Linear model is that of stages through which the design passes. Each of these stages is a collection of sub-activities. This is a classical view of the design process, widely reported in engineering text books. The linear model is straightforward, easy to comprehend and impress upon students, and it has been formalised in standards such as BS7000.

However it is a simplistic view of what happens in practice, and it is generally acknowledged that the design process is seldom as neat as the linear model suggests. Principal failures are identified by Raine (1998) and others as follow:

- Designers do not follow the sequence as the model suggests, and may rearrange the activities.
- Rework loops exist.

The issue of rework loops is addressed in the more complex models to be shown next. Provision is made for iterative loops to occur throughout, so that if for example

the detailed design runs into an insolvable technical problem then the concept design is reviewed. These loops are shown explicitly in some of the more sophisticated design models. The value of the linear model is in identifying the big picture and the broad stages of design.

1.2.2 **Models of intra-organisational influence**

The design process occurs within a business environment. Therefore a variety of other business factors (eg marketing, production, sales) influence the design process. There are several models of organisation process which are described next.

A model that explicitly studies the interaction of various internal company services is that of Hales (1994). The model includes various support services that are present in a typical design and production company. Some aspects of the model are illustrated in Figure 1.2.

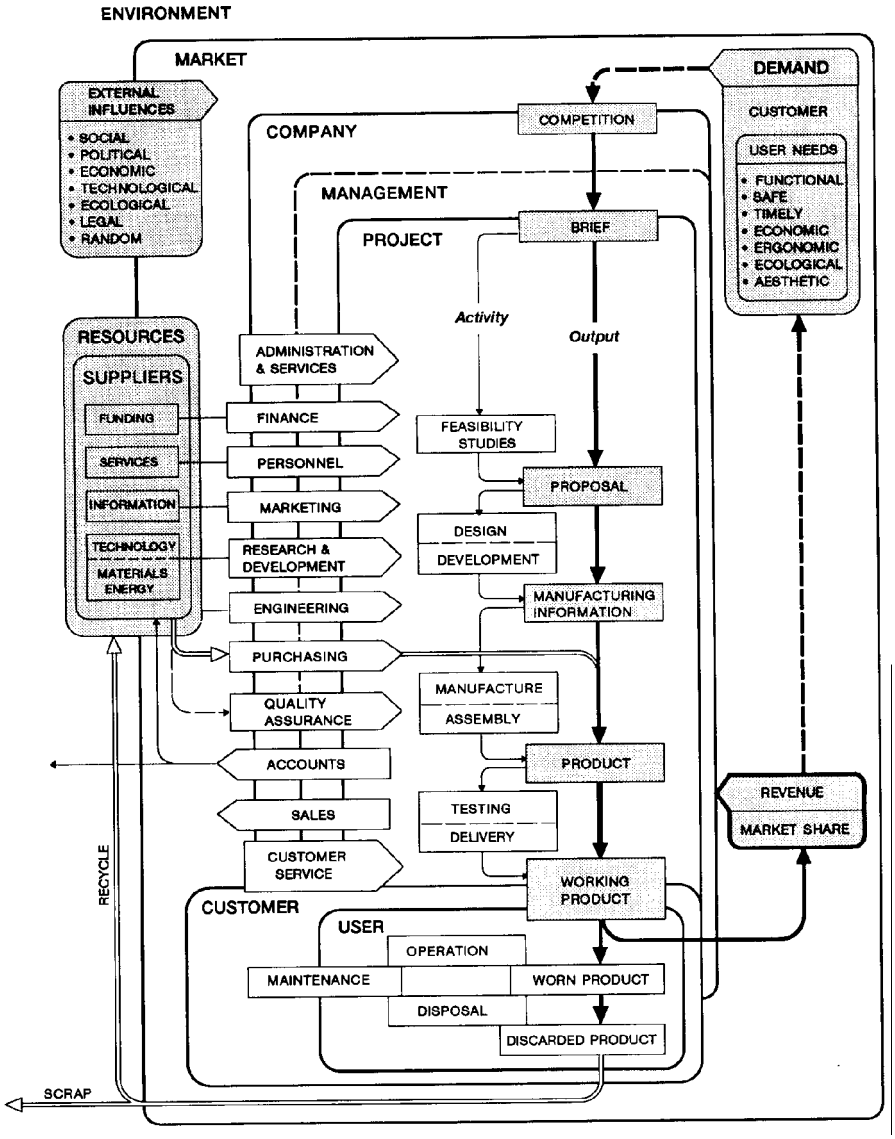


Figure 1.2: Hales' model of the design process. From Hales (1994).

Likewise Raine (1998) extended Pugh's concepts (Pugh, 1991) and created the model shown in Figure 1.3 specifically to deal with Customer focus, Exploration of design alternatives, Capabilities of CAD systems, and Concurrent engineering.

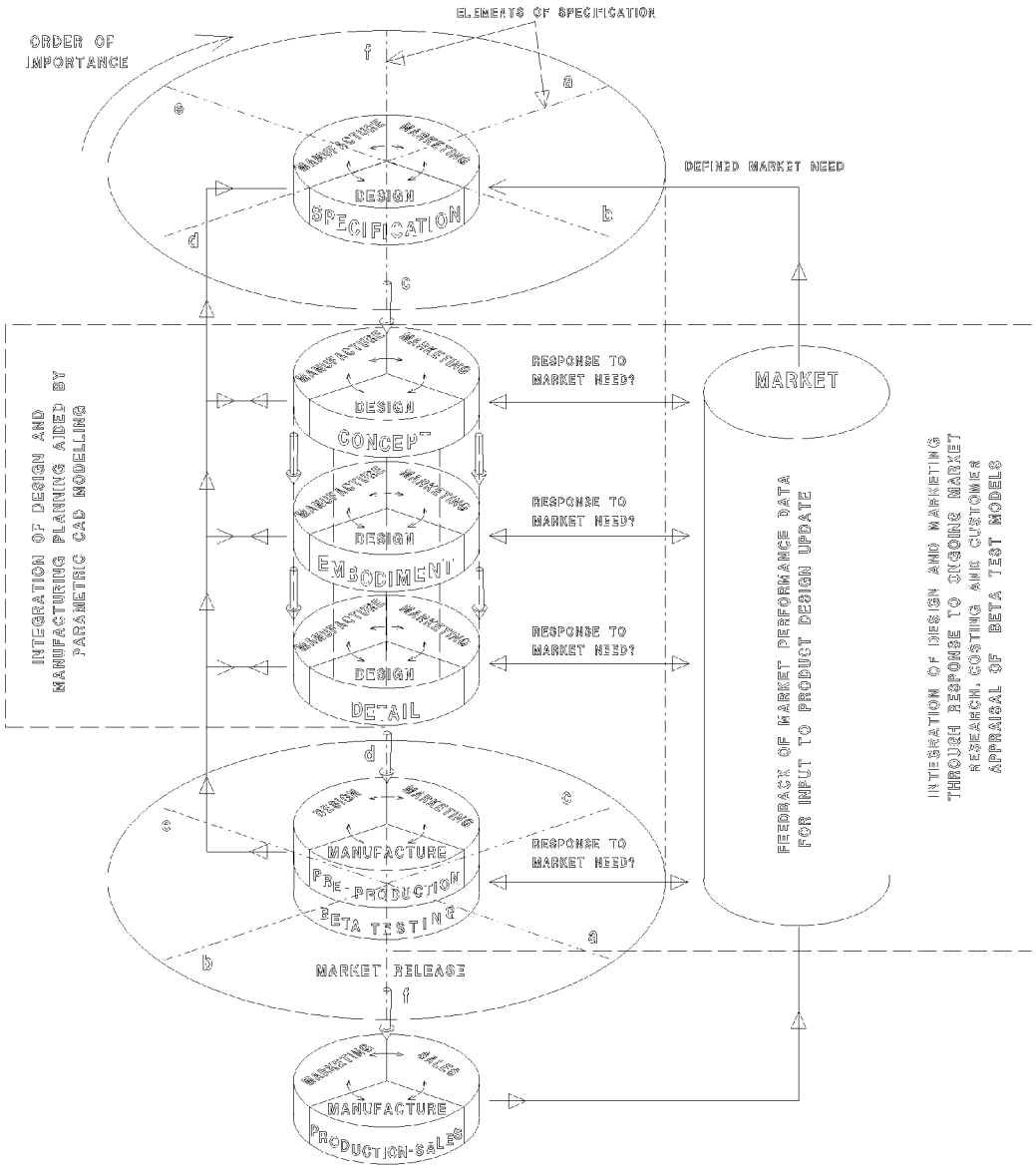


Figure 1.3: Raine's model of design activity. This model explores the integration of marketing and manufacturing with design activities. From Raine (1998).

Crisp (1986) elaborates on the viewpoint of corporate methodology, and identifies three primary streams in the design process, namely

- research and development
- production development

- promotion and marketing

Furthermore he sees these streams as interspersed, as shown in Figure 1.4.

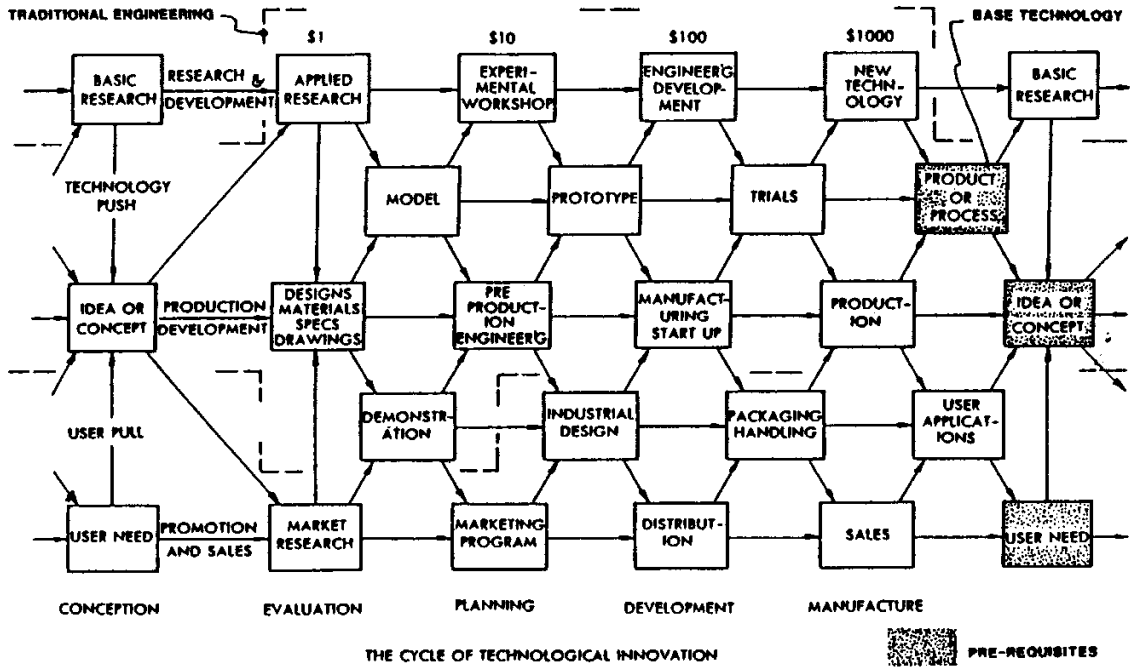


Figure 1.4: Model of innovation in design, from Crisp (1986)

These models explore how the design is influenced by other functions within a company, which is important in *concurrent engineering*. The intent is to get all relevant parties influencing the design earlier when the concepts are less fixed than they will be later[5]. Concurrent engineering has the potential to reduce developmental times, but introduces complexity into the management of the process (Medland, 1996). *Distributed design* problems (those where solutions are not independent) are often solved by a management decision giving priority to one part of the problem and requiring that the rest of the design compensate (Medland, 1996). There have been

[5]The emphasis in *concurrent engineering* is to reduce the product development times, so that a product can be brought to market sooner. This provides a competitive advantage, usually a necessary part of business success. Another term is *simultaneous engineering*. The way the development time is shortened is to integrate the essential business activities from early in the process, and to do the development activities in parallel rather than in series. To facilitate this, it is necessary to develop a corporate culture that emphases team work and communication across conventional departmental boundaries.

modifications to the principle of concurrent engineering, for example Krishnan et al (1995) propose reducing product development time by overlapping activities in a process they call "iterative overlapping". They use preliminary design information to initiate downstream activities, with design changes accommodated in subsequent iterations. Garcia (1994) motivates for the use of *risk* validation as a managed approach to product development, and a means for shortening the time to market.

Regardless of the method, concurrent engineering places particular demands on the management of the design process as the unknowns and hence risks are large. In comparison a linear approach to design is conservative with regard to risk in that opportunities are provided for unforeseen problems to be solved before commencing with the next stage of development.

1.2.3 **Models of individual designer's activities**

These are models not of the organisational process, but of the individual designer's process. Candy et al (1996) view the individual's design process as a spiral of repeated loops. Activities pass through quadrants (*ideas, solutions, evaluate, implement*) and change in strength with the passage of time, see Figure 1.5.

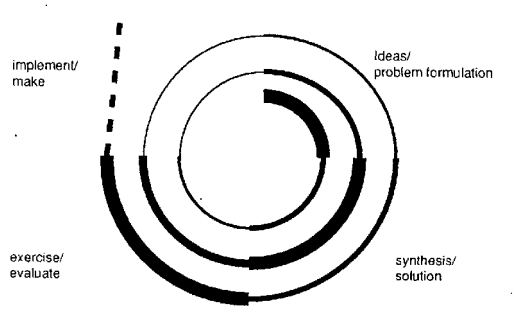


Figure 1.5: Spiral design process, according to Candy et al (1996)

There have also been studies into the method of design, with a particular focus being to elucidate the factors responsible for successful design. For example Wallace (1987) addressed the issue of information flow during design by having a *participant observer* record communication interchange during a design project.[6]

[6]The participant observer both contributes to the design and simultaneously observes it (Wallace, 1987). Successful implementation of this type of design research requires a special balance to be maintained between neutral observer status and active involvement. There is also the need to

1.2.4 **Phase diagrams**

Phase diagrams were developed by Hales (1994) and record the timing and the extent of activities undertaken. Figure 1.6 from Raine et al (1998) illustrates the principle. The various phases in the product development are shown, along with their timing duration and the amount of time used. Phase diagrams have similarities with project management tools such as *Gantt* charts.

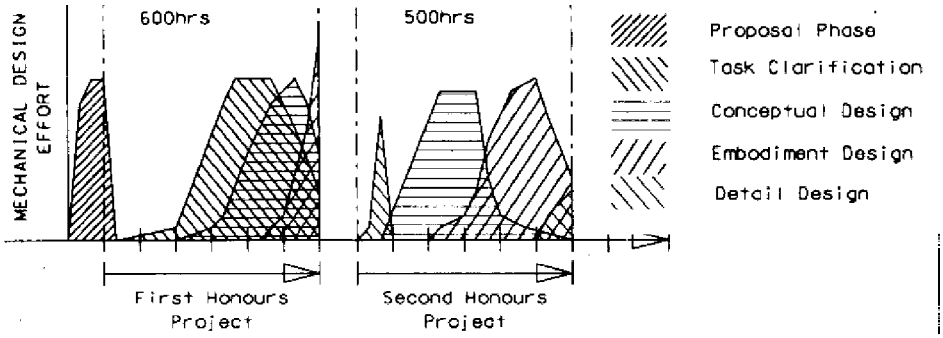


Figure 1.6: Phase diagram, modified from Raine et al (1998).

1.2.5 **Project management**

The *project management* methods are used to model the activities within a project. The project is decomposed to smaller tasks and these are represented on a time scale. A number of chart formats are possible, such as the *Gantt* chart (see Figure 1.7) which shows the activities against the time axis.

---

gain the confidence of the design team.

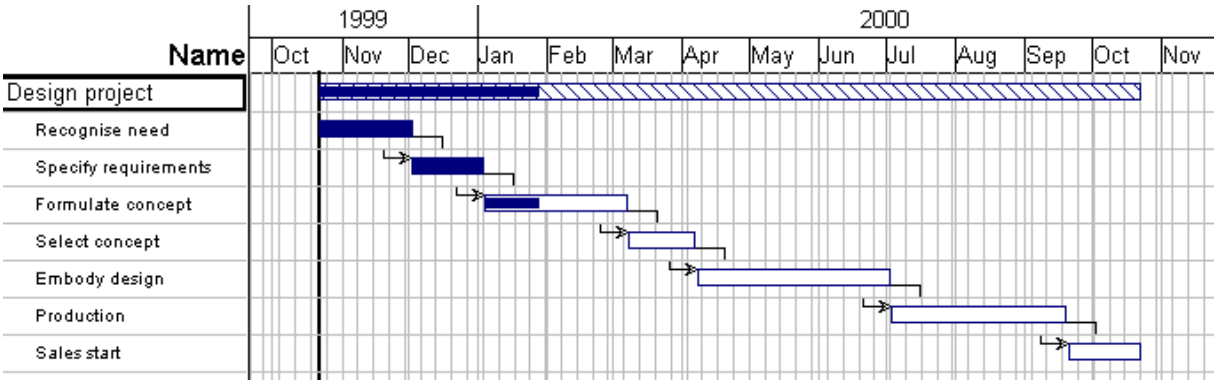


Figure 1.7: Gantt chart showing project plan

The Gantt chart is particularly valuable in managing a project as it shows the extent to which activities have been completed. Project evaluation and review technique (*PERT*) is used to model the uncertainty associated with schedule estimates in project management. The method uses three estimates of time for each task and combines them into a single time estimate.[7] However the PERT method is only an approximate probabilistic computation method.[8]

Yassine et al (1999) point out that the conventional project management methods are weak in the early design stages as they (1) ignore feedback loops (cyclic activities), (2) assume that the number of tasks and their duration are predictable, and (3) can only shorten the project by using more resources and cannot suggest concurrent activities. They believe that a "major problem in design process management is the existence of information cycles in the design plan", for example an earlier task requires information from a later task. *Integration definition* (*IDEF*) diagrams model the more complex activities that occur in concurrent engineering but Yassine et al feel they are very much more complex than Design Structure Matrix methods (described next).

[7]The estimates are used to determine a mean expected time, typically but not necessarily using a beta distribution for which the mean is t = (a+4b+c)/6 where a, b, c are the minimum, most likely, and maximum times respectively. The mean expected time is then used in the network as a deterministic value. The variance for each activity is v = ((c-a)/6)^2 for the beta distribution and the total project has a normal distribution (according to the central limit theorem) with variance as the sum of the variances of the activities on the critical path (Taylor, 1999, p463).

[8]More discussion on PERT and related network methods is provided in Chapter 2.

1.2.6 Extensions of project management: Design Structure Matrix and Signposting

Design Structure Matrix

The *Design Structure Matrix* (*DSM*) is a method that is used to analyse a system of tasks or other dependencies. Other terms are Dependency Structure Matrix, Problem Solving Matrix, or Design Precedence Matrix (MIT, 1999). The typical application for DSM is Project Management, where it is used to represent the tasks undertaken, and the order in which they are done. In this application it is frequently applied to study the design process. The advantage of DSM is that it is able to show the cyclic rework loops that often occur in the design process.

The Design Structure Matrix is illustrated by applying it to a portion of the design process model shown in Figure 1.8, producing Table 1.1.

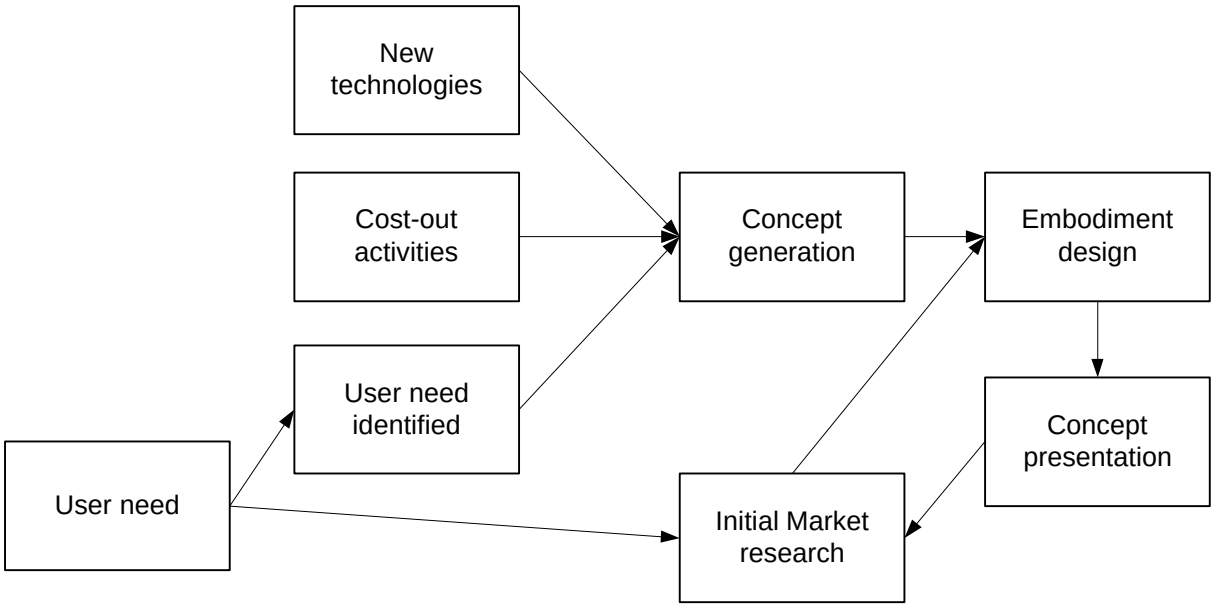


Figure 1.8: Model to illustrate DSM

		a	b	c	d	e	f	g	h
New technologies	a								
Cost-out activities	b								
User need	c								
User need identified	d			X					
Concept generation	e	X	X		X				
Embodiment design	f					X		+	
Initial Market Research	g			X					+
Concept Presentation	h						X		

Table 1.1: Design Structure Matrix for a fragment of the model shown in the previous figure. Precursor tasks are indicated by 'X' and return loops by '+'. The rows of the matrix reveal the input flows, and the columns show the output flows. Marks below the diagonal are for forward flow, while marks above the diagonal are feedback flows. Cells on the diagonal are blocked out as it is inappropriate to place marks here. For example "Concept generation (e)" requires that various precursor tasks (X) be completed: "New technologies(a)", "Cost-out activities (b)", and "User need identified (d)". Combined precursor and return loops are possible, eg "Embodiment design (f)" has a precursor task of "Concept generation (e)" and a feedback loop through "Initial market research (g)" which is via "Concept presentation (h)".

The DSM lists the subsystems and their interdependencies, including iteration loops. The feedback loops which are associated with rework activities and lengthen the development process, are simplified using *linear algebra* tools. The matrix is reordered to eliminate the feedback loops, or at least minimise them by moving them closer to the diagonal where fewer tasks will have to be reworked. Various '*partitioning algorithms'* have been developed for this purpose. [9]

The DSM method is also able to reveal *concurrent* (independent) tasks and the *critical path*. An alternative aim is to find subsets ('*clusters*') of the DSM with minimum interaction between the clusters. Consequently the DSM interactions are absorbed within the clusters. A cluster in this context could represent a particular

[9] DSM has been researched at various institutes, including Chalmers University (Sweden), NASA Langley Research Center (USA, developed the free software tool called DeMAID, which uses a genetic algorithm to optimise the matrix), Loughborough University (UK, applying it to schedule the design of buildings), the Problematics company (USA providing DSM management consultancy), University of Iowa (USA, clustering), University of Washington (USA, modelling the product development process), and VTT Building Technology (Finland, applying DSM to building construction processes).

design *team*, and the consequence of successful clustering would be that the interactions/ rework loops occurred as much as possible within the team, with minimised higher level interactions with other teams.

An extension of the methodology is Numerical DSM, where a number rather than a mark is used in the matrix. The number can provide additional information, eg the variability in output (Yassine and Falkenburg, 1999), or the probability of occurrence of the feedback loop (MIT, 1999). A further extension is to place two numbers in the matrix. For example Yassine et al (1999) use two measures to assess the severity of the cyclic dependency. The first measure is the variability: if the information in the feedback loop is highly predictable then the dependency is weak. The second measure is the sensitivity, which is how large the influence of the feedback loop may be: if the influence is small then the dependency is weak. They then multiply the two measures together to create a single measure of dependency, and use this to *tear*[10] the DSM. The sensitivity of the relationships and their variability have been used to enhance DSM and the prediction of time required for an activity (Yassine et al 1999, Carrascosa et al 1998). In addition they propose a risk management strategy, with weak dependencies being torn to independence, while management action such as overlapping is proposed for mid-strength dependencies and concurrency for strong dependencies. They use *overlapping* to refer to the use of preliminary design information in downstream tasks, before that design information is officially released. A related method is task redefinition, in which a task is split into smaller tasks so that the information can be released to downstream activities piecemeal. *Coupled tasks* are a pair of tasks that both output to the input of the other[11]. It may be possible to

[10]Tearing refers to eliminating those feedback loops that have weak dependencies.

[11]*Probabilistic reasoning* has been applied to determining the probable development time of a project (Carrascosa et al, 1998). The method used Design Structure Matrix with coupled tasks, modelled as a *Markov* chain of states. The accumulated development time was determined and displayed as a cumulative probability distribution. They used the model to perform a *sensitivity analysis*, i.e. to explore how changing the parameters affected to outcome. They acknowledge several limitations to the model: namely ignoring the learning effect (repeating a feedback loop may shorten the task time, or reduce the probability of further changes), that palliative adjustments (eg increased resources or reduced product features) are not modelled, and that only a few task can be handled due to computational limitations. This last point is a consequence of Markov analysis and the proliferation of states as more tasks are added to the model.

remove the coupling between tasks if the downstream task can accommodate variation in the output of the other task (Yassine and Falkenburg, 1999).

However the use of a simple multiplication function (with effectively equal weighting) is weakly motivated, especially considering that the entire tearing process hangs off the resulting single measure of dependency. On the other hand the method does represent a significant improvement over the plain DSM approach.

Signposting

Clarkson and Hamilton (2000) decompose a design problem into tasks and design parameters that need to be solved. Their premise is that design parameters are initially vague but converge to final values as the design progresses and other parameters firm up. They use *qualitative probability* (low, med, high) which they term 'confidence' to describe the maximum benefit expected from executing a task. They link the tasks together in a model that shows how the completion of one task affects the confidence (degree of finalisation) of other design parameters. They could then identify the most appropriate tasks to execute in order to be able to calculate a specific parameter to given confidence, and they call this process 'signposting'. Their approach is related to that of design structure matrix, and hence also to project management. They developed a software embodiment of the methodology.

1.2.7 **Design Science**

Design Science is the term for a logical framework which intends to contain and organise the complete knowledge for designing (Hubka, 1987; Hubka & Eder, 1996). The underlying concepts are a *technical system* (eg a machine) and a *technical process* (eg a manufacturing step). Hubka uses the term *anatomy* for the geometry of a technical system, and *organs* for its internal functional sub assemblies. The functional modelling concepts of decomposition and materials-energy-information are integrated into the framework. Hubka (1987) lists various design approaches, methods of concept design, and checklists for various stages of the design process.

The Design Science framework is an academic approach to design processes. While the checklists are likely to be useful prompts for a designer, some of the surrounding matrix of technical system, technical process, organs etc. may be difficult for designers to engage with. Eder (1998) asked why industry does not use Design Science, and offered the explanation that it will simply take time. Frost (1999) replied that industry already does use much of Design Science, since '*the models and methods which are described in Design Science are very often elaborate representations of what designers in industry have already been doing*' (p297). He also felt that '*the experiential knowledge of the seasoned designer in industry provides a faster avenue to a solution ... than would result from a deliberate and formal usage of the encyclopaedic fresh-sheet-of-paper methods which are commonly seen in Design Science*' (p302). He identifies the need to present Design Science material in ways that are relevant to Designers rather than Design Scientists. Such comments are relevant to the other design methodologies too. The challenge for us all is to continue to extend the academic methodologies of design, and also develop them into design tools that are sufficiently usable to be deployed on actual design tasks under realistic working conditions.

1.3 **Comment on design models**

The design models may seem conflicting. However, they have different viewpoints:

- Domains - the process as followed in one engineering industry is not necessarily the same as in another.
- Organisational and societal cultures - possibly some cultures are more strongly prescriptive about the design process than others.
- Content - some models are describing the activities that the designer undertakes (similar to a project management viewpoint), whereas other models describe the influences on the designer.

As Wallace (1987) points out, there are many ways of describing the design process, depending on the viewpoint taken, and design procedures should be applied flexibly.

What do we know about the design process and its tools?

Addressing the design process and its theory models, it is apparent that mechanical engineering design theory is a partial fit to actual design practice. To summarise using the review of Finger and Dixon (1989 a, b):

- Designers are not as systematic as the theory, and do not follow it precisely.
- Design theory is unrealistic about the orderliness of the process.
- Designers follow a single concept strategy at times. They reuse a familiar concept, fixing it as necessary rather than finding new alternatives.

Wallace (1987) discusses the obstacle of the non-repeatability of design, whereby it is impossible to precisely repeat a study on design method. This makes it difficult to come to generally applicable conclusions, or to systematically test design methods. He urges the development of systematic models of the design process, despite these limitations, such is the importance of the topic. He points out that even a small improvement in design process would have dramatic effects, and cautions that it is unnecessary and impractical to be too perfectionist in seeking an ideal design process. Furthermore he supports the development of empirically derived design methods as well as those of systematic origin.

Incomplete information at early design

The models do not explicitly address an important part of the early design stages, namely that designers often need to work with incomplete information, and make decisions under these conditions (Raine, 1998). Bartsch-Sporl and Bakhtari (1996) comment that complex design tasks contain requirements that are incomplete and contradictory, and therefore that the solution cannot be developed by a straight forward deterministic process. Instead the design is explored partially, and dead ends mean that it may be necessary to reconsider earlier decisions and rework their solutions. It may be impossible to simultaneously satisfy every requirement. The conflicting requirements have to be harmonised, perhaps by weakening the less important ones.

Failure on the designer's part

There are also some failures of the design process that are not due to modelling deficiencies but rather due to failure of the designer's diligence (Raine, 1998). Items here might include designing before a specification is complete, creating unnecessary complexity, fixating on a concept without considering alternatives, and failing to consider consequences or downstream requirements. Raine presents a fuller list.

Limitations due to personal factors

The design models do not address the effects that personal relationships and authority structures have on the process, and indeed it is difficult to see how such factors could be included. However, at least two factors are relatively well established:

- The designer's personal preferences and experience are an important factor. Anecdotal evidence suggests that designers often make heavy use of past experience when solving new design problems. No only so, but it has been observed that designers may stick to such a solution and put great effort into trying to make it work, rather than scrap the personal preference and seek other solutions (Raine, 1998).
- It is also impossible to dissect the designer out of the office culture, in that office politics and the leadership or hegemony of senior management in the business affect the design process and the integrity of the solutions. Raine (1998) found that the design process may be short circuited by autocratic management structures. The existence of formal and systematic design methods does not necessarily protect the design process from undue influence.

The effect of these intricate human factors is to make the design process somewhat more haphazard and irrational than the models might suggest.

Summary

The design process appears to be a variable mixture of chaotic and systematic creative processes, and the models provide little formal guidance on how to manage this balance. Neither do the models address the issue of the large uncertainties at early design stages. Ultimately the purpose of any design model is probably not so much to prescribe a methodology as to serve as a container in which may be stored the distilled knowledge of processes that improve the likelihood of design success. The design models address various perspectives of the process to various degrees of success.

1.4 **Design mechanism and constraint diagram**

For the purpose of systematically showing how the various design tools and research fit together, the author proposes a model of the design process that emphasises the mechanisms and constraints.[12] It uses integration definition zero (IDEF0)[13] notation. An advantage of this notation is that it permits inputs and outputs to be clearly distinguished from factors that influence the activities, so it may contain more information than the plain flow diagrams of other design models. Also, the IDEF0 notation readily permits zooming in to scrutinise internal details. At the top level the development process within an organisation is modelled as shown in Figure 1.9. This

[12]These diagrams are the intellectual property of the author but may be freely reproduced providing the source is acknowledged.

[13]Integration Definition (IDEF) is set of a diagramming standards for representing block diagrams (Addy and Simms, 1996). IDEF0 is used for modelling manufacturing and business processes, including business process re-engineering (BPR). Other IDEF standards cover information models and system dynamics. The block in an IDEF0 diagram represents an action, with arrows representing objects feeding into it or being produced by it. Four types of arrows are provided for: Inputs, Outputs, Controls, and Mechanisms (or Calls). The Inputs (if any) are transformed or even consumed by the function in the block, to produce one or more Outputs. Input arrows are always on the left of the block, and Outputs on the right. Controls, there is always at least one, ensure the output is correct. They are shown as arrows above the block. The Mechanisms (if any) that support the function are arrows under the block. The only outgoing arrow is that for Outputs (though a reversed form of Mechanism named Calls is possible too). Arrows are not so much flows or sequences but operating conditions (Kimbler, 1997). Diagrams may be arranged in a hierarchy, with a top diagram containing the overview and subsequent diagrams showing the details within each block. Numbering of the blocks, arrows and diagrams is defined in the IDEF standard.

model is at the same organisational level as those of Hales, Raine, and Crisp discussed above.

Explanation of model

The development process may start anywhere (this is not a flow diagram) but for purposes of explanation the discussion arbitrarily starts with the activity '*Recognise need to develop new product'.* The activity involves a design manager or product champion *initiating* the activity to *develop product* in response to a *directive* from those who *set the organisation priorities* or in response to how the customer perceives the potential *product worth*.

The *develop product* activity consumes working capital to produce a *product specification* within *constraints*. Some of the *key characteristics* of the product are potentially apparent after this stage. Various design *mechanisms* are used to support the process.

The *product specification* influences the manufacturing activity *produce product* which uses materials and labour to do so. Constraints exist, and the *projected sales volume* also influences the production activity. Concurrent engineering constraints are produced, and other key characteristics become evident as the final product emerges. The product is sold, generating *sales volume* and *nett income.*

The activity *set organisational priorities* uses nett income and financial reserves to develop new products. Constraints here include the organisation's *mission* and the *shareholder needs*. The strategic activity also responds to the success or failure of the anticipated *product specification* or the product itself. For example anticipated failure of the product to fully meet market needs affects the strategy taken by the organisation. The strategy-setting process can also produce *directives* which prompt new product development. The strategic activity involves management deciding how much *working capital* is to be released to the development. Design managers

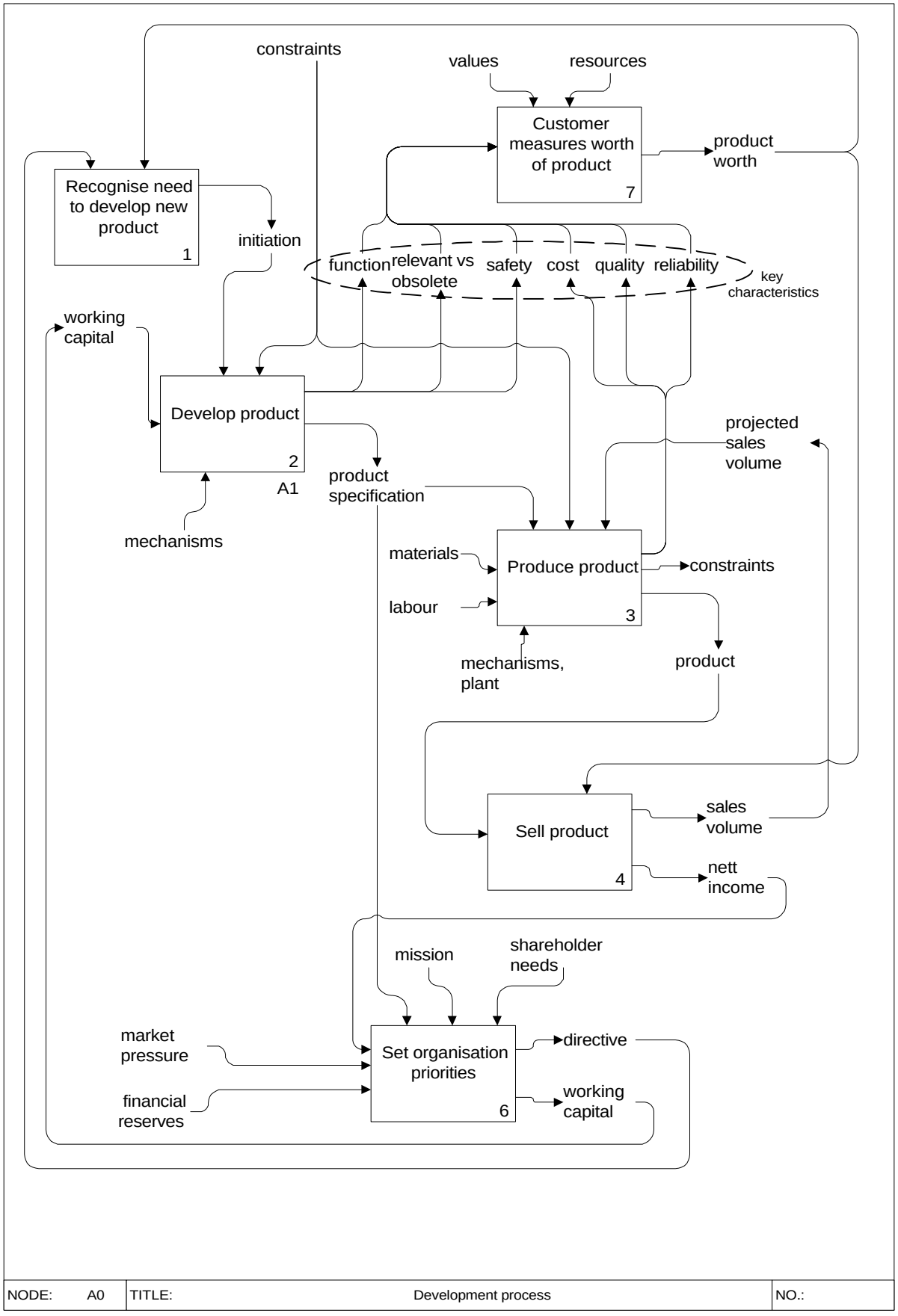


Figure 1.9: Design mechanism and constraint diagram at the top level.

naturally seek to maximise this resource as they see it affecting the success of the project. However there are other influences on the financial decision, such as the possible need to pay dividends (and thereby affect the share price and the capital from (a future) share listing) and the need to make savings to *reserves*. The influences discussed here are not exhaustive but simply illustrative of the financial constraints that affect resource allocation (sometimes also called *capital rationing*). Project funding is thus identified in the model as an important aspect of managing design, and one over which the design manager has limited influence.

Separately the *customer measures the worth of the product*, based on own *values* and *resource* constraints. This is a qualitative process depending on perceptions of a number of key characteristics of the product. These could include *function, relevance* (of style or function), *safety, cost, quality* and *reliability* among others. This relates also to the quality goal of 'delighting the customer'. Although these key characteristics might be quantified and known to the manufacturer, the customer usually has only tenuous information (though even that influences buying decisions).

The diagram thus shows how various parameters influence the design activities, without specifying precisely how those influences operate. Mathematical relationships might be inserted into the activity blocks to show how the output was dependent on the input and the control parameters. However to do so at the highly qualitative level of this model would be difficulty. Instead it is more useful to refine the inner qualitative workings of selected activities, and the *Develop product* activity is selected for this purpose.

Figure 1.10 shows four activities within the product development: *research market and define product specifications*, *design styling, design engineering details*, and *design manufacturing processes*. Each of these operates under constraints (arrows above block) using tools (arrows below block) to produce outputs. The arrangement of activities in the figure is illustrative rather than fixed. The model is not prescriptive as to the work breakdown structure in a project.

It is convenient to explain the diagram starting at the activity '*Research market and define product requirements'* at top left, though it should be noted that this is an influence rather than flow diagram so various activities can simultaneously be active. The '*Research*' activity takes any initial concept (includes ideas, market pressure from opposing products, benchmarking against other related products, and customer feedback) and produces specifications for the user interface (for styling) and product function specifications (for engineering design). The '*Research*' activity uses various mechanisms for market research (eg QFD, AHP, focus groups, and benchmarking) to achieve this, and is constrained by the availability of development funds, the willingness of the company and individuals to change, and the corporate culture.

The activity of '*Design styling*' takes the requirements for user interface and using various mechanisms (not detailed here), produces a styling model within the constraints of cost, ergonomics and styling conventions.

The activity '*Design engineering details'* produces engineering details (typically drawings) using various engineering design tools and mechanisms (eg computer aided design, finite element analysis). Design operates with the constraints of the styling model and the product specification that have been passed across. There are additional constraints for the engineering designer, such as those of cost, material strength, corrosion resistance, and reliability. Furthermore the manufacturing engineers concurrently apply constraints on the manufacturability of the product, which need to be accommodated at this stage.

The next task is to '*Design the manufacturing processes*', and this produces tooling and specifications for manufacturing processes from the engineering drawings. The activity uses mechanisms such as computer aided design (CAD), computer aided manufacture (CAM), computer numerical control (CNC), and mold flow analysis while operating under constraints of budget and manufacturability with existing resources and materials.

It is generally desirable that as many as possible of the above activities are

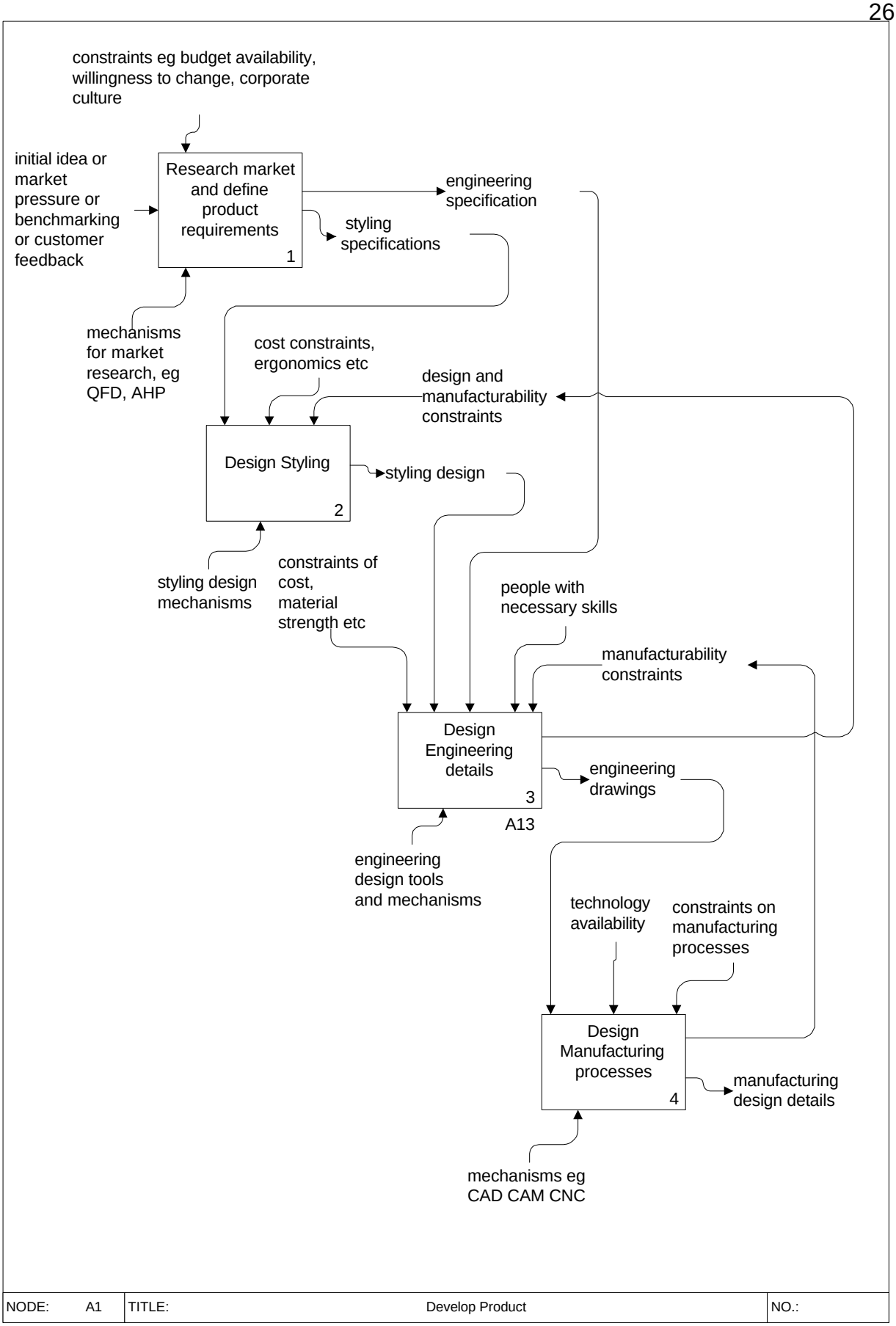


Figure 1.10: Design mechanism and constraint diagram for the activity of 'Develop product'.

happening concurrently, since this enables a quicker time to market. Concurrent engineering is accommodated in the model, in that each activity responds to constraints from others, and produces concurrent engineering constraints for others. However this is not mandatory as the model does not specify timing details. Likewise the model is nonspecific as regards the definitions of concept-, embodiment-, and detailed-design.

The philosophy of this model is that design is an augmentative process, i.e. that each design stage adds new value to the existing design, by using various mechanisms and taking various constraints into account. Importantly, the output of any one design stage is not necessarily the complete creative effort, as if the downstream activity was merely a non-creative technician task. Instead the output of one design activity may be a constraint (as opposed to an input) on the next, such that further creative activities are necessary to complete the bigger design.

The model also proposes a *generic design activity*,[14] which can be substituted in any of the design activities including *design styling, design engineering details*, and *design manufacturing processes*. It is proposed that all design activities, whether early or late in the cycle, have common attributes, namely:

- Finding a creative design solution to a problem that is:
  - (i) partially defined by some constraints, and
  - (ii) for which some partial existing concept may (or may not) exist.
- Assessing the proposed solution for suitability.
- Selecting a solution using some decision process.
- Implementing the solution to produce a detailed design.

[14]The *generic design* activity concept originated from observations of professional rivalry between Styling Designers and Design Engineers, despite the commonality of design tools. Why does each group so often consider theirs to be the fundamentally creative process that is critical to the success of the product, and hence that they are the 'true' designers? Perhaps the answer is that anyone perceives himself to be *the* designer if he is called on to produce something that did not exist before. However, when the Styling Designer delivers a concept, the Design Engineer feels that it is not so much a concept that could be built on as much as a set of constraints to work within. That is, one person's completed design output makes a limited contribution (if any) to the next person's design effort. The thread of design is creatively augmented at every stage from styling through to production and beyond. The Generic Design Activity explores the similarities in the types of mechanisms used at various stages.

The design is augmented at each design stage, in the sense that something new is added. A variety of mechanisms (methods and tools) are available for performing these activities. The Generic Design Activity is a standard building block that may be deployed at various stages in the design cycle, with different mechanisms operative. Any number of Generic Design Activities may be connected together to represent the total design cycle.

The graphical representation of the Generic Design Activity in the context of Engineering design is shown in Figure 1.11. There are five main sub-component activities which are connected to each other and to other Design Activities by inputs, outputs, and constraints.

The Generic Design Activity is a model of a reusable design activity, that can be positioned anywhere in the design process and at as many locations as design occurs. The Generic Design Activity addresses primarily the design activities and does not attempt to model all the other enterprise activities.

Problem definition (specification) is a part of most design processes and is included at the previous level diagram. It is also implied in the Generic Design Activity as constraints such as ‘prescribed inventive constraints’, ‘concurrent engineering constraints’, ‘prescribed assessment constraints’, and ‘reasonably anticipated constraints (professional judgement)’. Such constraints may arise either from an imposed and premeditated specification, and/or from requirements that arise during the design process. The model does not attempt to stipulate where the constraints arise, or prescribe whether they be formally established before design or in an ad-hoc basis during design. [15]

[15]While it is considered good practice to create a specification before design starts, there is anecdotal evidence that not all designers in industry operate that way. In some cases (eg artistic design or consumer product design) it may be difficult to set either a comprehensive or an objective specification before design begins. Therefore though an activity of problem definition occurs, it is not necessarily always a formal engineering specification but could instead be a marketing research activity resulting in a relatively soft and amorphous definition. Either or both types of problem definition may be used in a design project, and the model accommodates both without taking a prescriptive approach.

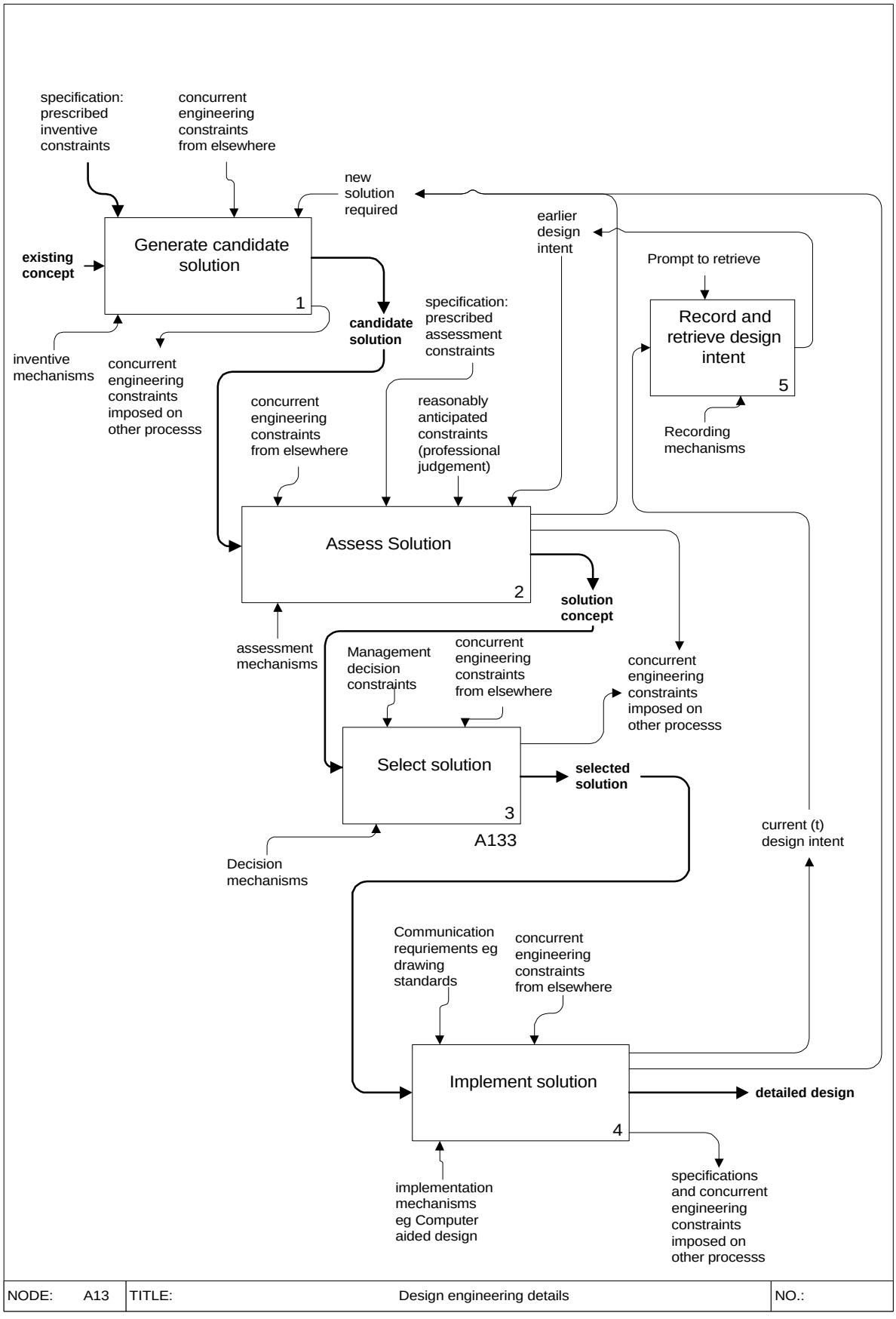


Figure 1.11: Generic Design Activity in the engineering design context.

The internal activities of the Generic Design Activity are as follow:

(1) Generate candidate solution

At this stage the designer (human or artificial intelligence) takes the existing concept (if any), and uses various inventive mechanisms to create a *candidate solution* within prescribed constraints on how the solution should perform. These constraints could originate from upstream or downstream activities. Concurrent engineering constraints from other design activities may be included here too. At styling and early design the upstream inventive constraints are provided by market survey, focus group, quality function deployment (*QFD*), analytical hierarchy process (*AHP*), and conjoint analysis. At later stages the preceding stages provide the constraints.

If the design is an incremental improvement on an existing design, then a well-defined input concept exists. Another way to produce an existing concept is through experience of a prior similar solution from another domain (eg natural analogy). In the more general case of innovative design that breaks new ground, there may be no existing concept and the solution must be created *ex nihilo.* Even if there is a Design activity preceding the one under scrutiny, it is not necessarily safe to assume that the design output of the predecessor is the existing concept for the current Design activity. It may be more relevant as a constraint. For example, the completed styling design of a dishwasher provides no existing concept to the design of the wash pump, only a possible constraint on the size.

The inventive mechanisms include human serendipity, brainstorming, systematic idea generation, catalogue methodologies, theory of inventive problem solving (*TIPS*/*TRIZ*), morphological analysis, genetic algorithms (and related shape annealing), grammars, expert systems, and artificial intelligence methods. Such mechanisms are described in further detail below.

The output of the Creative solution generation sub-component is a candidate solution. This is next tested to see whether it complies with other requirements. It is possible that several solutions may be considered simultaneously and be in various

positions within the Design activity, and several Design activities in a larger system may all be active.

(2) Assess Solution

The candidate solution is next assessed for validity. Possible assessment mechanisms include focus groups (at early styling design), system simulation (throughout engineering design), functional modelling, grammars, bond graphs, feature based modelling, risk assessment (including qualitative, hazard and operability study, quantitative, fault tree analysis, and failure mode effects analysis), Monte Carlo, sensitivity analysis, design of experiments and loss functions.

The assessment constraints may originate from prescribed upstream constraints, for example manufacturing design may receive geometric tolerances from an upstream detailed design. The activity also includes reasonably anticipated constraints, which are *professional judgement* constraints of endogenous origin[16]. In cases of incremental design, or where design is reworked, the proposed solution must also not violate earlier *design intent*. If the proposed solution fails the assessments, then either the feedback loop initiates the generation of a new solution or the design proceeds with an imperfect solution.

The primary output from this sub component is a solution concept. This adds value to the original candidate solution, either by clarifying or adding new information. The solution concept is then input to the next sub component. Also output are preliminary constraints for concurrent engineering (cf the ‘overlapping’ concept in DSM). Although the design has not yet been firmed up, there may still be sufficient information for other up- and down-stream activities to begin their processes.

[16]The designer would be negligent if he failed to assess those things that he could reasonably be expected to be able to anticipate. Health, safety and liability constraints may also be applicable in this category.

(3) Select solution

At this sub component a decision is made whether or not to accept the input solution concept. Decision mechanisms include conflict resolution and decision analysis (including belief networks, influence diagrams, and decision trees). The activity is constrained by concurrent engineering requirements from elsewhere in the systems (up- or down-stream). However it may not always be possible to simultaneously satisfy all the constraints in a system, and for this reason a management decision might be made to give priority to one activity and require the others to compensate. The action of selecting a solution freezes part of the design and imposes constraints on other design activities.

(4) Implement solution

Here the selected solution is consolidated into a detailed design, for example by producing working drawings or models which provide sufficient detail of the design solution. Communication requirements determine the format and extent of the consolidation process. Other outputs are a record of current design intent, and constraints on other activities.

If at any stage the design fails then the concurrent engineering constraints provide the feedback loop to initiate re-examination of previous activities.

(5) Record and retrieve design intent

This sub-component exists to record the design intent and to retrieve it and feed it forward as a constraint to solution assessment. Recording mechanisms include notebook, grammar, and proprietary software. The design intent is useful in preventing redesigned solutions from unintentionally violating some requirement. Recording the rationale for a design is one part of the task, but it is essential that something initiates the retrieval process where appropriate.

The activities shown here are generic, and have been set up with an individual design task in mind, such as may be performed by an individual designer or small team. The system boundary is therefore around such a group. A separate Design

Activity could model each design team in each of the various stages of the design cycle. Importantly, the Generic Design Activity concept is flexible as to the composition of a team, so it does not matter how the distinctions are drawn between stying, engineering and manufacturing design.

1.5 **Quality of information**

The discussion now turns to the *Select solution* activity of the Generic Design Activity, the internal operation of which is now explored in Figure 1.12.

The model is briefly described as follows. The core activity is to '*Make a decision*', which is done under *management decision constraints* and using various *decision mechanisms,* and produces a *selected solution.* The figure elaborates on the management decision constraints and decision mechanisms. Making a decision on the *selected concept* has consequences, which may require that the decision be adjusted as those consequences develop. If the decision is made early enough (*time permitting*) then there is greater opportunity for such adjustment. Any adjustment is constrained by *uncontrolled and unknown factors* though other *controllable factors or tuning parameters* may be available for manipulation.

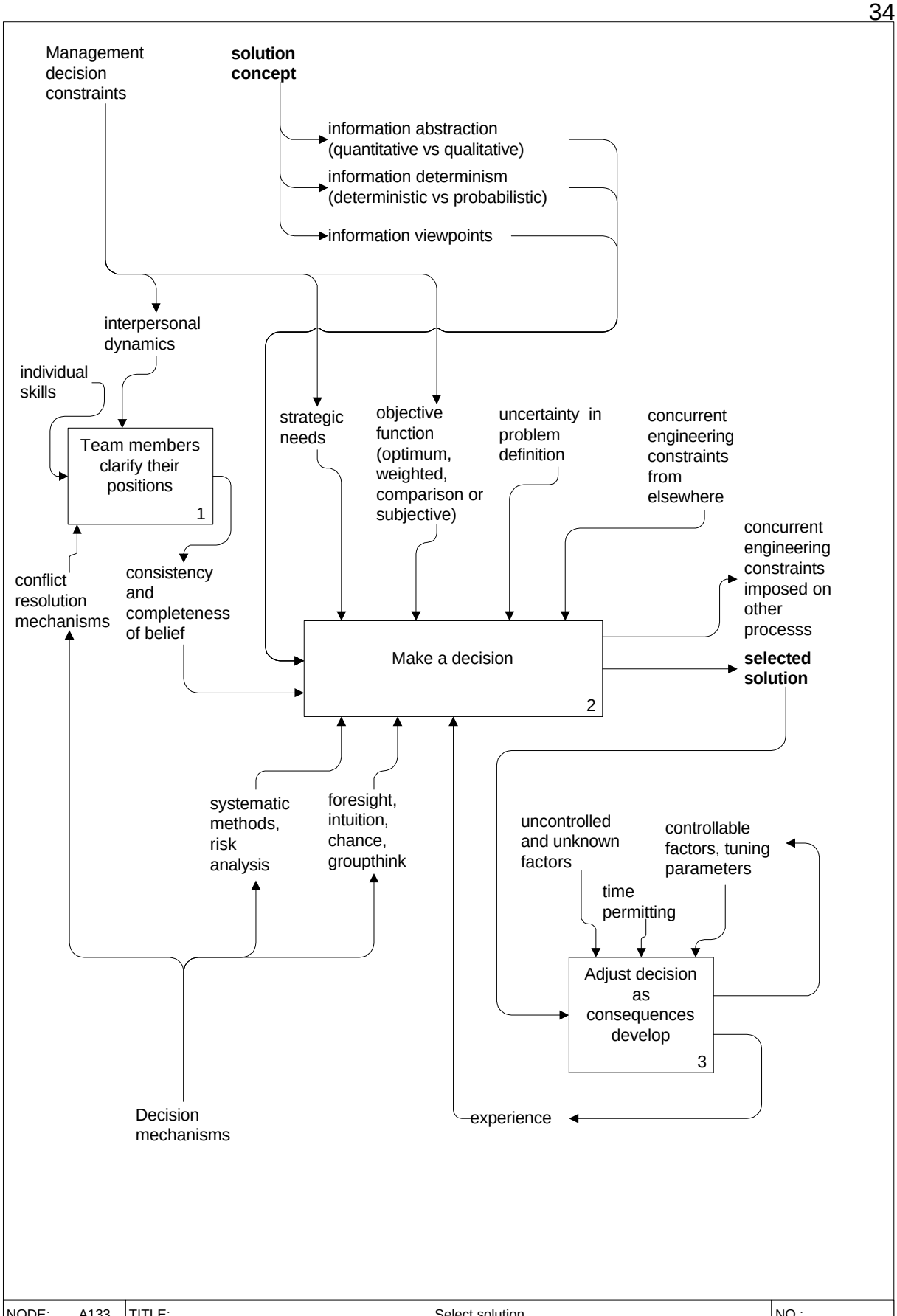


Figure 1.12: Model of the decision process, in particular the influences on a designer or design manager who is selecting a solution.

For the decision process to be robust it is necessary that the solution concept contains quality information. The information results from a previous assessment (analysis) activity. Extending on the work of Ullman and D'Ambrosio (1995)[17] the information in a concept may be classified according to several independent attributes:

- *Uncertainty of analysis (completeness of knowledge)* refers to the degree with which knowledge is held about how variables determine an outcome. Ideally there would be mathematical relationships defining the outcome in terms of input variables. However knowledge is not always available to this extent. Instead it is sometimes necessary to rely on rules and logical expressions, e.g. those processed by expert systems. Sometimes only a lesser degree of knowledge is available, in the form of expert opinion. This attribute is mentioned for completeness and is not illustrated in Figure 1.12 as it is implied as part of the *Assess solution* activity of Figure 1.11.
- *Information abstraction*: Information may be quantitative (ratio or interval scales) or qualitative (ordinal or nominal). This is discussed in further detail in Chapter 6.
- *Process variability (Determinism)*: Information may be deterministic or probabilistic (stochastic). There are various degrees of probabilistic information ranging from single point probability values, multiple point probabilities, moments, and full probability distributions.
- *Viewpoint:* Information may cover one, some or all key viewpoints.
- *Belief:* Information from team members may be more or less consistency with each other, and individuals may have more or less completeness in their beliefs (confidence of opinion).

This classification states that the strongest information, and therefore that which most assists the decision process, is that which is quantitative, represented by probability

[17] Their classification was for engineering decision problems, and while some of the attributes are used here (abstraction, determinism, and belief) the rest are more appropriate for other activities. For example their objective function appears in the current model as a constraint on the decision rather than a type of information. The chief difference between their classification and the current one is that theirs does not distinguish between factors that are input, output, constraints or mechanisms for the decision process.

distributions over multiple viewpoints, and on which the whole design team agrees.

This classification completes the exploration of the design process. Later, in the discussion chapter, the Design for System Integrity methodology is compared to this same classification.

1.6 **Conclusions**

Existing models of design process include the linear model, models of design as part of other organisational processes, phase diagrams, project management models (and related design structure matrix). However design practice is not as orderly as the models suggest. The design process may depart from the theory for various reasons, one of which is incomplete information.

Another view of the design process has been developed, termed the Generic Design Activity. Its context in the larger organisational processes has also been described in a design mechanism and constraint diagram. The intent in doing so is to provide a framework for systematically grouping[18] the design tools in the next chapter.

[18]There is no intent at this time to attempt to simulate a design project through the system.

Chapter 2

Literature on design mechanisms and constraints

This chapter reviews the literature on methods for assisting the design process, with particularly emphasis on the capabilities of these methods to operate in the early design stages where uncertainty is high. The Generic Design Activity is used as a framework for placing the methods into perspective.

2.1 **Inventive mechanisms**

It is beyond dispute that the early stages of design have a vital effect on the downstream consequences of the design process. For example Brady and Juster (1996) maintain that “future efficiency gains in new product introduction will depend in part upon the availability of tools that aid the conceptual design process”, and many other authors have expressed similar sentiments. The conceptual design stage is where the bulk of the project costs is committed, and therefore the solution space needs to be thoroughly explored (Kersten, 1996). Ideally all design alternatives would be evaluated, but this is often impractical. This section reviews the inventive mechanisms, some of which are depicted in Figure 2.1.

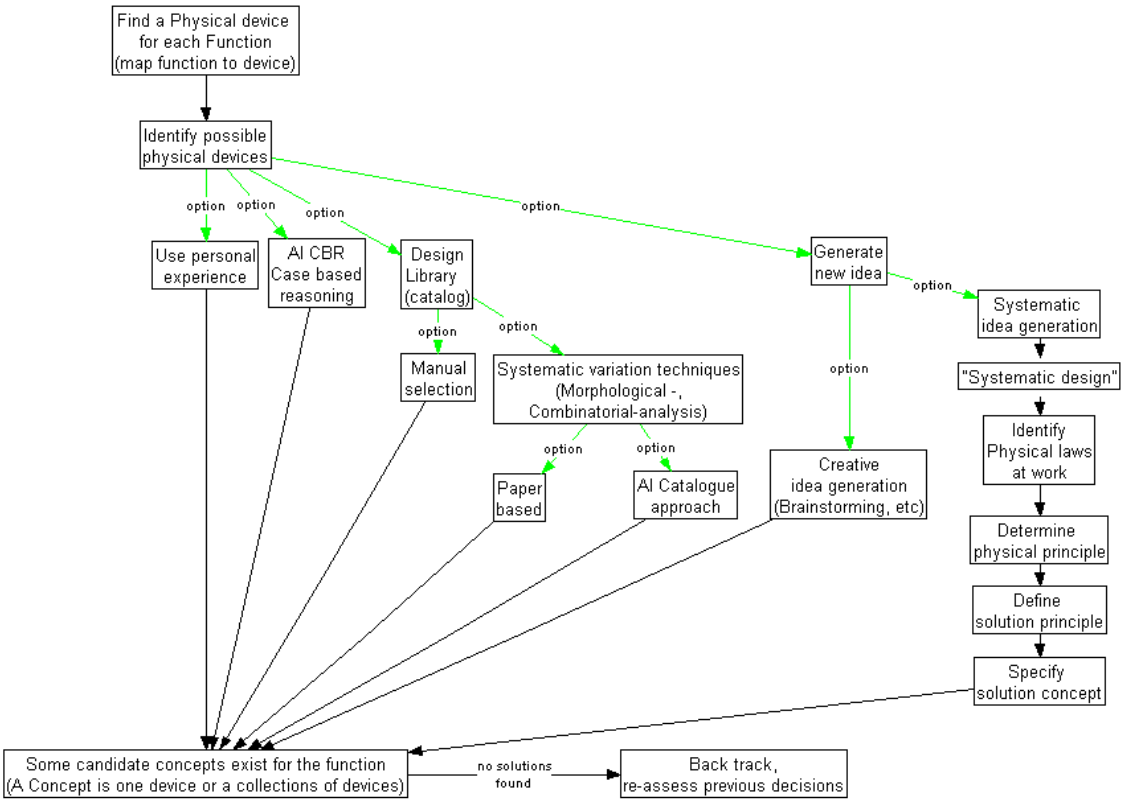


Figure 2.1: Inventive mechanisms

2.1.1 Human based inventive processes

The inventive stage involves identifying possible physical devices to meet the required function. The default method for this is through personal experience. The human designer who has previous exposure to a successful solution may have reason to use that solution again if it appears to fit the new problem. However there is the risk that designers may to reuse past solutions even though they may not be appropriate or optimal for the task at hand. An alternative to using personal experience to solve a design problem is to find and use a proven solution technology from other people in or connected to the organisation, patent searches, academic and commercial literature, and commercial catalogues. If the human designer is unable to find an existing solution, then there is still the possibility of generating a new idea. There are several methods that are in use for creative idea generation, such as brainstorming. These methods rely on a team approach to the problem, and particularly on building association between different ideas.

2.1.2 Functional decomposition

Many authors such as Pahl & Beitz (1988) and Finger and Dixon (1989 a, b) differentiate design according to the level of ingenuity required:

- *Original design or Fundamental ingenuity*
- *Adaptive design or Radical application*
- *Variant design or Incremental improvement*

The systematic inventive methods all use decomposition processes. *Functional decomposition* involves decomposing the specification into functions at lower levels. A variety of methods may then be used to seek a mechanism for each of the decomposed functions, see Figure 2.2. The solution to the system problem is then a synthesis of the sub-solutions. If it is impossible to decompose the functional requirements then the problem definition may lack completeness or the problem is one of *distributed design* and cannot be decomposed. Most classic design strategies, including those implemented in morphological analysis and many artificial intelligence

tools, assume that the design problem can be decomposed, i.e. that the functional requirements are independent of each other.[19] For example Suh (1998) postulates an '*Axiomatic design theory*' with two axioms: (1) maintain the independence of the functional requirements, and (2) minimise the information content of the design. Functional requirements are mapped to the physical domain to give design parameters (which can be constraints) and then process variables and finally physical parts. All functional decomposition methods rely heavily on the problem being decomposable in the first place.

[19]The term *functional modelling* is used by some authors for the process of modelling a design problem by its decomposed sub functions. However there is some ambiguity in the usage. Some authors use *function* for what might otherwise be called *design intent*, that is the function that the designer intended the machine to perform. Another usage of *function* is the physical *behaviour* or performance of a system, usually in the context of a design that has been modelled as a network of black boxes with *functional relationships* between the inputs and outputs of those boxes to produce a quantitative simulation of machine behaviour. The term *function* is also sometimes loosely used to refer to different viewpoints in the sense of *performance function*, *manufacturing function*, etc.

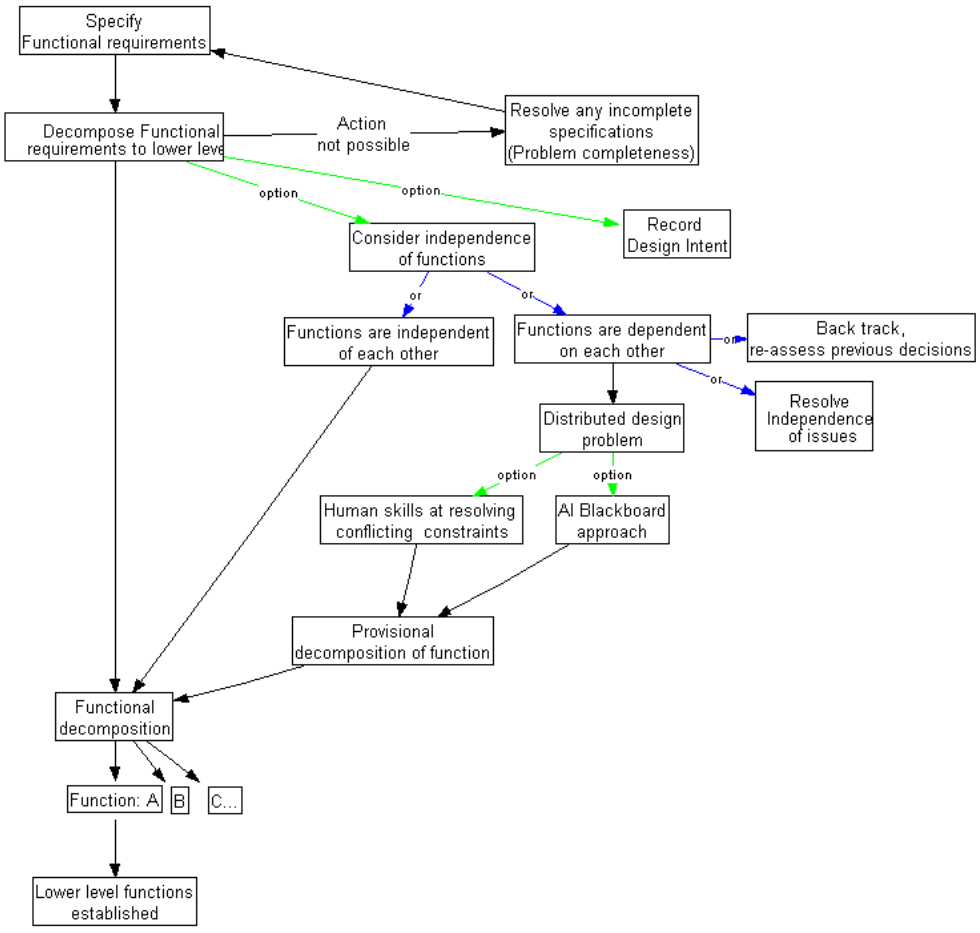


Figure 2.2: Functional decomposition process

2.1.3 **Mapping processes for inventiveness**

Once the primary problem has been decomposed into sub functions, the designer seeks to *map* the individual functions to specific physical devices. The difficulties are that (1) the function may be provided by more than one device, and (2) a physical device often provides more than one function. More problematically, many of the functional requirements in a design may be difficult to specify quantitatively. The link between function and form is incompletely understood, especially for conceptual design. Mapping from desired function to design description can currently only be made for certain domains such as mechanisms (Finger & Dixon, 1989b). These constraints aside, there are several mapping methods, which are described below.

2.1.4 Catalogue processes

Mapping functions to physical devices has been done using a predefined list of physical devices and their functions, known as a *catalogue*. The catalogue contains design components specific to the domain. Potentially there may be more than one catalogue component that can satisfy the functional requirement, and a selection must be made between them. Automated systems have been proposed and developed around *neural networks* and *case based reasoning*, but within relatively tightly defined domains. The catalogue method is a modular approach to design, and for its success it requires that the design problem be sufficiently well decomposed. Catalogues are commercially available, describing physical principles. For example "IM-Phenomenon" (Fletcher, 1998) is a database of working principles with descriptions of the principles involved. This system represents principles, and it is entirely driven by the human user. The system does not attempt to synthesise principles together to meet functional requirements.

Clibbon et al (1996) represent the logic of conceptual design as a *mapping* between two layers, Figure 2.3. One layer consists of descriptions of object models, and the other of the requirements. Their domain of application is *motor vehicle* packaging (of occupants, engine etc.). They use a set of primitive constructs to represent the model and its requirements (constructs include *element-of, component-of, power-set-of, product-set-of, union-of, intersection-of, and pair-of).* Their intention is to develop the system in a *Prolog*-like system. Catalogue

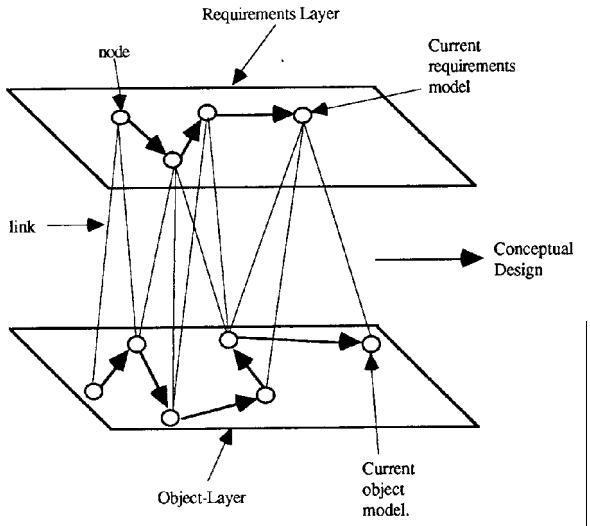


Figure 2.3: Multi-layered logic, from Clibbon et al (1996)

systems have also been combined with decision analysis systems (Bradley and Agogino, 1991).

Kersten (1996) describes a system called “Modessa”, which assists the design process. It is not an artificial intelligence application, nor does it contain rules. Instead it is a *relational database* and a graphical user interface. The domain is materials handling, and packaging in particular. In use the design problem is decomposed by the designer to a functional level. The database contains solutions to functional problems, and the designer selects concepts from the database. The system can also search for design alternatives given the functional requirements. Assessment is by the designer providing a score for each of mechanical performance and electrical performance. Once the concept is selected at this level, then the process repeats at lower levels of function. Weighting tables are used to score the alternatives. The use of weighting factors forces the designers to find objective arguments. Kersten maintains that the method “produces better structured design documentation that allows a reader to understand the design decisions made”. The system uses small pictures (“morphological overviews”) to illustrate the principles of various candidate solutions. Kersten calls the process morphological design but this is potentially confusing as the method does not perform combinatorial morphological analysis. The use of *weights* is a common approach in many design tools that seek to assess qualitative[20] parameters. The approach is vulnerable in that the weights can be difficult to defend.

2.1.5 **Systematic idea generation**

Pahl and Beitz (1988) propose a *systematic design theory* for the establishment of function structure. The initial action in the process is to clarify objectives and boundary conditions, and then to decompose the overall required function into sub-functions. Then each sub-function is considered in turn: one of a number of

[20]The interested reader is referred to Chapter 6 for a description of how the terms quantitative and qualitative are used in this work.

physical effects is selected and developed into a solution concept. With more detail added it becomes a solution and if it is suitable then the process is repeated with other sub-functions. Finally the solutions for the individual sub-functions are combined to give the solution for the macro design problem. The process is illustrated in Figure 2.4.

The systematic method accommodates intuitive thought processes as well as *discursive* (logical) methods.

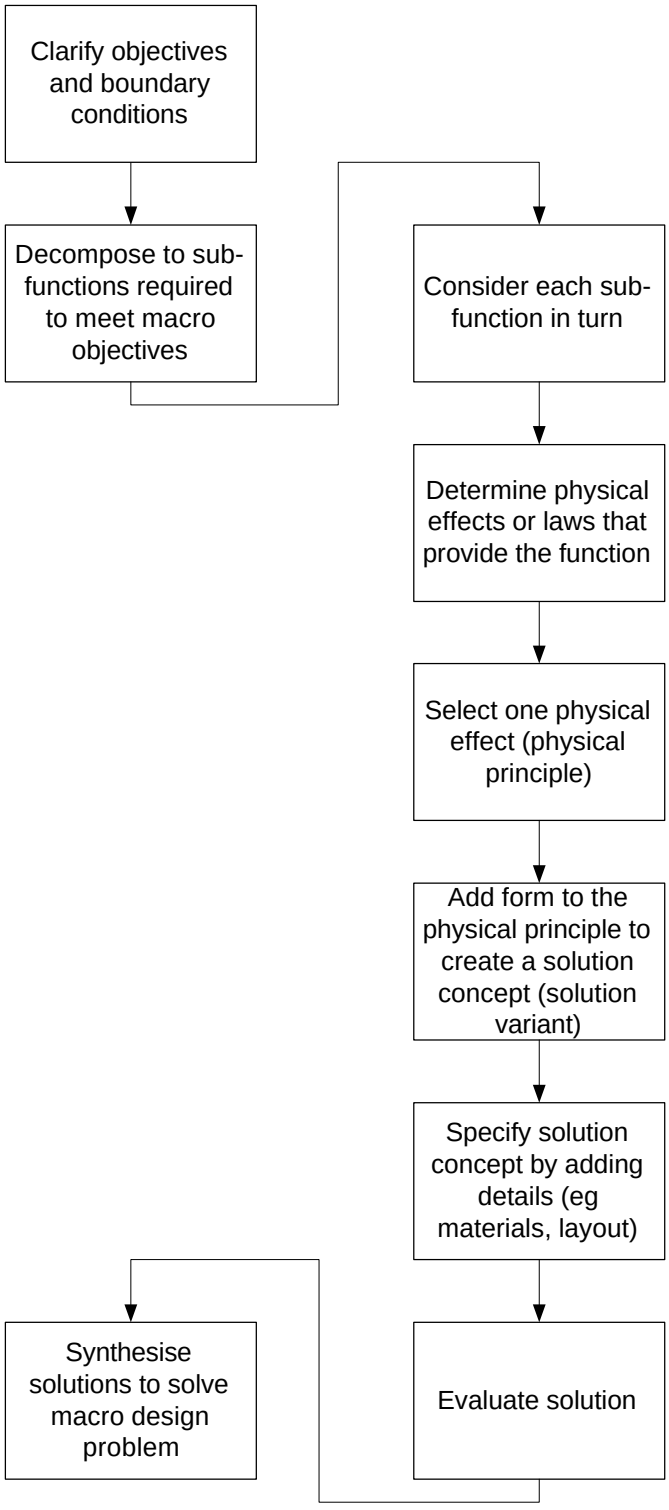


Figure 2.4: Systematic idea generation process.

Comment on the systematic method

Systematic approaches have been used to create various models of the design process. There are numerous models of systematic design according to the authors concerned, and different terms and classifications exist. The German engineering industry has a strong history of systematic approach to design, and has produced many of the pioneers in this field. However the systematic method is not without controversy, since some

see it as a prescriptive method. Its intention is possibly not so much to prescribe the creative process as to facilitate it, and Pahl and Beitz (1988) give the purposes as inter alia:

- encourage problem solving approaches,
- encourage ingenuity,
- reduce the dependency of success on chance,
- minimise the errors in a design, and
- facilitate the application of known solutions.

Another example of the method is given by Ishii and Tomiyama (1996) who describe a knowledge base named QPAS that reasons about physical phenomena. The system involves functional decomposition into sub-functions, and then selection of mechanisms that provide those functions, followed by synthesis of engineering mechanisms. The system accepts different operating states of the machine and seeks to derive suitable structures to meet these requirements. It does this by referring to the knowledge base of physical phenomena. The database includes entities (eg shaft, gear), relations (eg on, above, connected, fixed), physical phenomena (motion, heat flow, evaporation, friction), attributes (position, temperature, mass), and physical properties (elastic, magnetised). One of the attributes is *mode*, which may be set at values such as *on* or *off*. This determines the state of the entity. This may be used to represent a desired behaviour or *state transition* (operating condition). The designer is able to build new physical features into the model. The system tries to find physical features that meet all the operating states of the machine. If this is not possible then the different operating states have to be met with multiple features. The selection of one physical feature may require another feature that was not noticed at first (eg a ball screw needs a motor). In this case the designer adds the function to the requirements. The system adds unexpected physical phenomena to the design. These are side effects that detract from the required function. The system produces a graph of state transitions, one for the required functions, and another for the predicted behaviour.

Yet another example of a systematic approach is that of ‘Structured planning’ (Owen, 1993). It uses functional decomposition to obtain a tree-based ‘function structure’. The functions are then scrutinised for intended operation as well as failure mode in a process not unlike failure modes effect and analysis. Speculative solutions are proposed and then assessed for how many functions they relate to. The interactions between functions[21] for a given solution are assessed on a scale and transformed into a graph. The graph is then analysed for clusters of connected (related) functions to support creative synthesis by human designers. The examples given by Owen (1993) are more styling rather than engineering design problems, and it is interesting to see such a systematic approach to creativity in this area which is otherwise more usually known for its adoption of the more chaotic creative approaches. Many designers (both styling and engineering) would perceive the ‘structured planning’ method as formidable, and Owen acknowledges the ‘considerable job of assessment’ (Owen, 1993, p101) necessary to obtain the function graph. This could be a limitation, as could be the dependence on functional decomposition.

To some extent all the systematic methods, and related methods such as ‘Design Science’ (Hubka, 1987), are arduous and elaborate. They consequently tend to have limited relevance to working designers (Frost, 1999) so other supporting systems are required to increase their accessibility.

2.1.6 **Morphological analysis**

Morphological analysis is a method to generate and select design alternatives. It involves decomposing the problem into sub problems, and then generating many solutions for each subproblem. Multiple system solutions are then found by combining lower level solutions. These combinations may be created by random selection. Finally the multiple system solutions may be evaluated and a selection made (Finger and Dixon, 1989 a).

[21] This is ‘the degree to which the two functions are related’ (Owen, 1993, p99).

Pahl and Beitz (1988) use the term *systematic variation* for the process[22]. Other terms are *configuration trees*, and *combinatorial* generation of solutions. The method relies on the assumption that subproblems are independent of each other. The process has been applied to well-defined design tasks where there are limited combinations.

An example of the combinatorial approach would be Hague et al (1996) who describe the proposed development of the “Co-Designer” tool. They anticipate that the user will input functional requirements for the new product, and then the system will *generate solution variants* by combinations from a knowledge base. The designer could also browse and select solutions manually. They propose using an algorithm to optimise the solutions. They intend that machine learning be supported, in the form of solved designs stored as weighted *decision trees*. These may be used to assess the consequences of a design decision, and also reused in other designs.

There are several examples of artificial intelligence applications of morphological analysis.

2.1.7 **Concept invention with Artificial Intelligence**

Artificial intelligence attempts to simulate human reasoning and intelligence, using software[23]. There are several artificial intelligence tools of which expert systems are possibly the best known.

---

[22]The approach presented by Pahl and Beitz (1988) is:

- Definition of the task, clarify requirements.
- Abstracting to identify the essential problems
- Search systematically for known physical effects and solution elements.
- Systematically vary the possible elements. Combine them into possible solutions.
- Identify shortcomings of the various solutions, and try to reduce them.
- Select optimum solution.

[23]Artificial intelligence concerns the automation of knowledge based tasks, in a manner which is essentially a computer programming method, with some similarities to database software and conventional programming languages.

There are three types of knowledge used in computer applications, and these are:

- Declarative knowledge, which describes the state of a condition, is stored and manipulated by database software.
- Algorithmic knowledge describes a procedure or flow chart, and may be written as a routine in a programming language.
- Conditional knowledge describes logical conditions, experience, and special case knowledge.

Artificial intelligence differs from other computer programming languages in that it deals with conditional knowledge. It is often impossible to express *conditional knowledge* adequately in terms of a flow chart, and is therefore difficult to implement with conventional procedural programming languages (including object oriented programming languages).

Artificial intelligence methods have been developed to represent and process conditional logic and a number of sub-disciplines are included:

- Logic based programming languages, such as Lisp and Prolog, which were the origins of artificial intelligence. These languages exist in several versions from different vendors.
- Expert systems which implement knowledge through conditional rules.
- Artificial neural networks and genetic algorithms, which are programs that avoid the need to explicitly define rules, but which learn knowledge in an inductive fashion.
- Computer-based design. This works for specific domains where the problem is well defined, and is less effective when information is incomplete or abstract or inconsistent. Optimisation methods may be used, but require that all aspects be defined quantitatively. Assemblies may be built up from standard components that have been selected from a library or catalogue (Finger and Dixon, 1989 a, b).

Artificial intelligence requires the capture of knowledge, and its formulation into rules that the system can implement. Ishii and Tomiyama (1996) give key concepts in *knowledge engineering* as model building, simulation, model-based reasoning, model

validation, and model modification. The knowledge may be acquired from a human expert through interviews, and consists of a set of *if..then* statements. In some cases an expert can state the possible results and the contributory factors. In other cases it may be possible to convert an existing database into a set of rules (Ignizio, 1991). If the artificial intelligence application is *backward chaining* (eg diagnostic), then a *decision tree* may be used as the knowledge base. A decision tree is another form of knowledge representation, and is easily converted into a set of *production* rules. In many cases there are different versions of a decision tree possible, depending on the viewpoint. In a fully defined deterministic situation a tree with as few levels as possible is desirable, since it reduces the number of user responses required. However it may be unwise to have the bare minimum, as extra rules are required to deal with partial information (Ignizio, 1991).

2.1.7.1 Logic based programming languages

The advantages of *logic based languages* such as *Prolog* are in performing logical deduction and proof. These languages represent *first order predicate logic* (Biondo, 1990) in terms of individual statements. Artificial intelligence applications may be written in Prolog-like formulations, but these are clumsy tools for developing an expert system. This is due to a fundamental limitation of Prolog and similar logical languages, namely that the order of rules is important. This forces the programmer to organise the knowledge in terms of logical flow charts, an activity that is more cumbersome and error prone than alternatives. Furthermore the number of axioms in the system increases as more facts and rules are added, and there is no reliable way to retract an axiom. The knowledge is therefore said to be monotonic in the way it grows. The advantages of Prolog-type languages are in performing logical deduction and proof, rather than in forming the basis for an expert system.

2.1.7.2 Expert systems

Expert systems are rule-based programs. They have a *database* of information, and a set of *rules* as to the interpretation of that data. These rules may cover declarative, algorithmic or conditional knowledge. They also have an *inference engine* that provides the capability to process the data and rules. Most expert systems provide both backward- and forward-chaining.

- *Backward Chaining* is a diagnostic problem solving procedure that begins with a statement and searches backwards through the rules and database to establish the truth of the statement.
- *Forward Chaining* is a predictive problem solving procedure that begins with the rules and database, and works forwards to a find conclusions based on these.

Most Expert systems now use *production systems*. These are rule based, and use similar expressions to the logic based programming languages, but the critical difference is that the order of the rules is immaterial. The advantage of production systems is that they do not depend on a user-programmed system to check all the rules, and thereby permit the rules to be added without premeditated order. Therefore knowledge may be incrementally acquired and added as rules, without having to reorganise the sequence of program execution. Importantly, there is also the means to retract facts that are no longer needed. Production systems still need a method to check the rules though, and they do this automatically, by using a selection algorithm such as the *Rete algorithm*.[24]

[24]Examples of production system architectures are OPS5, ART (Inference Corp), CLIPS (NASA), ART-IM (Inference Corp), Eclipse (Haley). CLIPS was developed by NASA. It is public domain software, available for a relatively minor cost. There is a large base of users in various disciplines, although there is no formal product support from NASA. ART-IM is commercially available from Inference Corp, although at a greater cost than CLIPS. However the vendor does provide customer support. Eclipse is available from Haley Enterprise for a relatively minor cost, and customer support is provided by the vendor.

There are several additional features and enhancements to expert systems. One of these is *case-based reasoning* (*CBR*). It uses past solutions to suggest solutions to current problems, and thereby provides the means to accumulate learning and experience in artificial intelligence.[25]

Expert systems are typically developed on existing *shells*. These shells are provided by the vendor with an inference mechanism (backward chaining, forward chaining, or both), user interface, interface to other programs and databases, and sometimes automatic generation of web pages. There are many expert system shells, some free and others available at a cost. Not all features are available in every shell. The knowledge has to be entered in a particular proprietary format. This means that expert system shells are generally incompatible with each other. The shells may be considered a type of programming language. The interaction between the user and the expert system is typically through a menu or graphical interface, the quality of which varies between the systems. The interested reader is referred to Giarratano and Riley (1994) who discuss the principles of expert systems, with particular reference to CLIPS.

A *blackboard system* is a collection of multiple expert systems that all work on the same model[26]. Each expert system has its own knowledge base and inference engine, and it posts its results on the blackboard where the other expert systems can use the information. The individual expert systems work independently, and can use different inference methods. They are opportunistic in that they watch the blackboard and seize the opportunity to add solutions when possible. A control module decides which expert system (also referred to as a *knowledge source*) may make the next move. The benefits of Blackboards are given by Biondo (1990) as the ability to handle specialised and distinct knowledge, and integrate such information. Furthermore they accommodate sparse knowledge or data, as well as continuous

[25]CBR has been applied in some expert systems, relevant products being Easy Reasoner (Haley), CBR Express (Inference Corp), Remind (Cognitive Systems).

[26]Applications include HEARSAY (II & III), HASP, AGE, BB1, HANNIBAL, TRICERO.

data (i.e. data changing with time). However, they are relatively weak in dealing with uncertainty.

Frames are artificial intelligence structures for holding default knowledge (Biondo, 1990). A frame refers to an object (eg a flowering plant), and consists of a number of slots, each of which describes a particular attribute of the object (eg a slot for flower colour, another for flower shape, leaf shape etc.). The whole domain (plants) may be divided into increasingly detailed levels of frames and slots. Many but not necessarily all expert systems use frames and slots. Frames introduce the concept of *inheritance*, whereby an object automatically inherits the attributes of the parent object, so that if a plant has been defined to have leaves, so a flowering plant also has leaves. This is an economical representation of knowledge since it avoids the necessity to define every attribute for every object in the knowledge base.[27]

Candy et al (1996) demonstrate a design support system for vehicle design, and systems packaging in particular (LUTCHI). Their approach is to provide *rules* that are specific to the domain of study. They permit the designer to modify existing rules and add others, in order to extend the knowledge base. Their domain is the packaging of *automotive* systems (occupants, engine, fuel tank...) in a motor car layout. The system permits the designer to place objects graphically over an image (eg scanned in sketch). Notes may also be tagged to items in the knowledge base. The system then uses the rule base to provide feedback about the design.

2.1.7.3 **Expert system case study: CAPE**

Cunningham and Smart (1993) describe the implementation of a system for computer-aided parts *estimation*. Termed CAPE, the system is an expert system that generates production plans and cost estimates for a given automotive component.

[27]The concept of *inheritance* was developed in a programming language called “Smalltalk”, and this has subsequently led to the development of *object oriented programming* (OOP).

The system was developed by Ford Motor Company. It is an ambitious system to capture the knowledge of estimating experts.[28] Benefits of CAPE are given as:

- Capture expert knowledge about manufacturing processes as a human estimator might not be sufficiently expert in all the manufacturing processes, i.e. supports less experienced estimators.
- Training tool for estimators.
- Reduced time to produce detailed estimates of component cost.
- Reduce time from concept to customer.
- Quicker evaluation of alternative production processes
- Understand and control manufacturing costs.
- Provide supporting detail to help negotiation of price with supplier.
- Design for cost effectiveness and simultaneous engineering between designers and cost estimators.

The user defines a part[29] and the CAPE system then generates alternative process plans, and costs each of them in detail. It also seeks potential risks ("proximity to physical constraints such as maximum machine power rating"), and opportunities ("measures that could be taken to reduce cost" eg changing shift times).

It is interesting that the system uses Lisp macros instead of the rules that would be expected of a conventional expert system application.[30] Whether these macros have the order insensitivity property of production rules is not detailed. However, in common with expert systems, CASE separates knowledge from processing.

---

[28]Cost estimating of parts is a major business activity for an automotive manufacturer. The estimators are production engineers who are responsible for determining the cost of components in preparation for negotiating the contract with a supplier. The functions that the human estimator performs are to consider the operations, machines and materials that make a part. Alternative methods need to be considered so as to balance the piece and investment costs.

[29]Part definition is in terms of production volume, source country (takes into account different labour rates etc), and the structure of the assembly. Assembly processes are specified. At the component level the estimator describes the component features and material. The geometry of machined features is described. With formed parts (molded, pressed, cast), a qualitative measure of shape complexity is used. Surface coating specifications are also entered.

[30]The system is implemented in Common *Lisp* Object System (CLOS) using object oriented modelling. The interface is textual and graphic, and based on *ART* windows from Inference Inc. The database is ART SQL, and is linked to corporate databases.

Knowledge (such as of which features can be made by which processes), is held in local databases. The system provides heuristic (rule based) searches to find a solution. It does not use a combinatorial approach as this was believed to be infeasible.

Validation was a difficult problem in the CAPE system (Cunningham and Smart, 1993) because the actual costs of a component are not known for comparison with system predicted costs. Instead they used a battery of existing estimates (of human origin) and used these to validate the system.[31]

2.1.7.4 Concept design using Grammars, Genetic Algorithms, and Simulated annealing

Grammars are rules for the creation or description of a class of objects (Finger and Dixon, 1989b). They have been applied to engineering design, where they have been used to create geometry (*shape grammars*). The term *parse* is used for the creation of an object by the grammar, and has similar meaning to creating an *instance* in object oriented programming.

Andersson (1994) proposes a grammar (vocabulary) for the conceptual phase of mechanical design. He asserts that a shape grammar alone is insufficient for conceptual design, since the geometry involved is rough or absent. Conceptual design is concerned instead with engineering concepts and non-geometric information. He uses a verb and noun to describe functions.

---

[31]The CAPE project represents a major investment in software development. It took several years and 16 full time staff. The project was undertaken in three stages. The first phase studied air cleaners only, and was intended as a feasibility study. The second phase aimed to prove the feasibility of covering all manufacturing processes. Phase three improved the robustness of the system, and deployed it. A major task was integrating the expert system with the corporate *databases*, a task which "has been as great as the development of the expert system itself"(Cunningham and Smart, 1993) p45.

A *genetic algorithm* simulates a natural selection process. It takes a large population of prospective solutions and assesses them according to constraints provided by the user. Each individual solution is made up of components (*genes*). These solutions are usually generated from a combinatorial process. Each solution *(individual*) is subjected to an evaluation criterion, called a *fitness factor.* Solutions are scored on how well they pass the evaluation. Solutions with higher scores (*qualifications*) are given a greater probability of making it to the next round (greater chance for *reproduction*). Reproduction is done by two means:

(1) *mutation:* randomly changing one of the components (*gene*) in the solution, or

(2) *crossover:* selecting two solutions and swapping some components between them.

These techniques introduce variety into the solutions. The genetic algorithm process continues until all the constraints are satisfied, or a group of solutions dominates. An advantage of genetic algorithms is that good solution components tend to be successful and spread into other solutions too. Disadvantages of genetic algorithms are firstly that there is no guarantee that the best solution will be found. Secondly, and perhaps more importantly, genetic algorithms require formal specification of the constraint (*selection*) criteria, and are sensitive to these constraints. Different constraints will produce different results, and in addition the results are sensitive to where crossover points are made (Santillan-Gutierrez and Wright, 1996). Genetic algorithms have been applied to catalogue type design in the embodiment stage, where attributes of the components are well defined (take discrete values), and constraints are also easily specified (Santillan-Gutierrez and Wright, 1996).

Hudson and Parmee (1996) discuss the application of *genetic algorithms* to conceptual design. Their proposed methodology is to generate designs by random combinations from a *design grammar*. These designs are then subjected to evaluation and given a probability of success. The more successful concepts are used to breed the next generation. The design grammar takes solutions from a design *catalogue* of components. These components are retrieved according to their function.

Grammars may be comprehensive and thereby help the designer recall solutions that may have been forgotten. However a combinatorial synthesis can result in so many solutions that a human designer is overwhelmed. Genetic algorithms help eliminate the useless solutions. The difficulty with genetic algorithms is the need to define quantitative evaluation criteria. The designer provides the evaluation criteria, but these are often only known qualitatively. In addition there is the need to judge between the relative importance of criteria. Hudson and Parmee (1996) propose avoiding this problem by ranking solutions according to dominance (Pareto optimisation). An alternatively approach used by Pearce and Cowley (1995) is to use *fuzzy theory* to produce quantitative constraints from qualitative information. Likewise Farag et al (1998) use fuzzy theory as a front end to *neural networks*. However fuzzy theory addresses only part of the qualitative problem as it requires that the qualitative parameters be ordered. The qualitative parameters are transformed by the fuzzy set to a number, and this number is then processed further. The transformation lacks robustness and invalidates the fuzzy approach when the qualitative parameter is weakly ordered (see Scott and Antonsson, 1999). Also there are qualitative parameters that cannot be ordered at all, and for which the fuzzy approach cannot begin to be applied.

Simulated annealing (also called *recursive annealing*) is a method for creating and evaluating a large number of solutions. The system generates possible solutions by randomly replacing devices in a grammar string. Then it applies the evaluation function (provided by the designer) to assess the solution. The method explores the design space (that is all the combinations of devices known to the system) and finds a local maximum for the evaluation function. This will be a solution, but not necessarily an optimal one. The algorithm is able to explore deeper levels of the design and then recur to higher levels as necessary.

Schmidt and Cagan (1993) propose a grammatical model for conceptual design, with mapping of function to physical form using recursive annealing. The grammar is used to generate designs. It consists of an energy classification scheme. A number of physical devices are included (gear pair, belt drive, power screw, shaft, rack & pinion, flywheel, cam, torsion spring, linkage, compressive spring, electric motor, solenoid, and ratchet). Each of these is classified on two counts: the activation energy (rotational, translational or electrical) and the produced energy (continuous energy flow, reciprocating energy flow, continuous or reciprocating energy flow). The grammar imposes the requirement that compatible energy flows must occur between components. In use the designer provides the machine specifications and selects an evaluation function. The recursive annealing algorithm generates solutions from the grammar, and assess them with the evaluation function to find a solution. The string grammar used is suitable for strings of devices in series, but not parallel. More expressive grammars are necessary to create realistic machines (Schmidt and Cagan, 1993).

Santillan-Gutierrez and Wright (1996) propose the use of *genetic algorithms* at the end of the conceptual stage. They divide product knowledge into seven categories: geometric, assembly, materials, manufacturing, function, human factors, marketing. They concentrated on assembly (not specific to one product domain), and sought to minimise the number of parts in a design.[32] Their proposed process is as follows:

- functional decomposition
- collect all known devices that can fulfill the functions
- generate combinations of components,
- evaluate by suitable constraint criteria
- use genetic algorithms
- redefine constraints, becoming more selective, eg start by eliminating physically impossible solutions, and repeat the process

[32]In particular they used Boothroyd's classifications scheme for assembly operations. Proposed software is Genesis (from Geffenstrette).

The approach of Tang (1996) is based on a blackboard, using genetic algorithms and simulated annealing. It is intended to accommodate incomplete and inconsistent design requirements, to provide justifications for selections, identify areas of difficulty, support the exploration of the design from different directions, and include facilities for learning. The work is based on components from:

- Blackboard system, *(Assumption-based Truth Maintained Blackboard ATMB)*, Edinburgh University)
- Functional synthesis system (FUNCSION, Cambridge University), which searches for elements that match given functional requirements. It uses qualitative reasoning.
- Design embodiment tool, (CADET, Cambridge University), which builds a product library from generic functional components, and applies genetic algorithm or shape annealing. The weakness of CADET is seen as the inability to provide suitable model definitions (for functional synthesis, design embodiment, or kinematic analysis), and some limitations in the definitions of constraints for the genetic algorithm process.

Tang proposes to integrate parts of these systems.

Semantic networks are directed graphs. As such they use graph theory, nodes and links between the nodes. Semantic networks are useful in describing the relationships between objects. The principles of semantic networks have been extended into *frames* and into *decision theory*. Frames are an artificial intelligence concept, while decision theory is a topic peripheral to artificial intelligence.

2.1.7.5 Concept design using experience, heuristics and case based reasoning

Case based reasoning (CBR) refers to the solution of problems based on preceding cases of some similarity. It is an artificial intelligence method for applying past experience to the current design problem. It involves identifying elements in the current problem that match elements in a case base. The case base is a database of

past solved elements. The major activities in applying CBR are setting up the case base, identifying relevant cases to retrieve, and adapting them to the current problem. For its effectiveness the method requires that the current problem be decomposed into smaller problems, so that the database of past solutions may be searched.

Bartsch-Sporl and Bakhtari (1996) use case based reasoning on the grounds that it does not rely on the completeness or consistency of the domain. Their artefact is called *FABEL*, and its domain is air conditioning of buildings. In use the system selects past cases that are most similar to the current task. These are presented to the designer, who adapts them by hand. A partly implemented component of the system is the means to provide the user with suggestions for improvements, point out conflicts, and evaluate the merits of alternatives. The system is implemented on an *object* basis, with cases being fully instantiated objects, and schemata are objects with partially filled slots.

Non-monotonic reasoning is an artificial intelligence method that attempts to deal with changing knowledge. As new information is learned by the system it invalidates old information. While the approach is intuitively attractive for modelling human learning experiences, the methods require further development (Biondo, 1990).

2.1.7.6 **Neural networks**

Neural networks are artificial intelligence simulation systems with the somewhat unusual characteristic of not requiring rules. The neurons (‘*perceptrons*’) are software or hardware building blocks, each of which has multiple inputs but only one output. The output will fire if certain input signals are present. Simplistically this firing could be when the summation of the inputs exceeds a certain threshold. An activation function (typically a sigmoid function like the logistic) may be provided on the output to limit that output to a determined amplitude range. A number of neurons may be connected into a network (essentially a *directed graph*), and the system can then be

given certain inputs and will respond with an output. Generally the inputs, eg the pixels in an image, are each fed into one neuron in the input row of the network, which then produces a smaller range of outputs, eg the identity of the shape. The network could be a simple single-layer of neurons, or multiple neurons. Feedback loops (with delays) are possible.

Neural networks have to be trained, and in *supervised learning* the network is provided with test cases with inputs and known outputs. During the test case the internal weight for each neuron is modified so as to minimise the difference between the simulation and the real result, typically using the sum of squared errors as the criterion. The *back-propagation algorithm* is commonly involved to achieve supervised learning. It involves applying the input signal vector to the network and passing it forward to produce a set of outputs. The error is then propagated back through the model difference to readjust the neuron weights. Another mode is *reinforcement learning* in which there is no external teacher, and the network learns by interacting with its environment and observing the consequences of its decision.

Advantages of neural networks are non-linearity of response, provision of a mapping between inputs and outputs (without requiring explicit rules, though still requiring explicit knowledge representation), adaptability to new data, fault tolerance (the removal of some neurons does not destroy all resolving power of the network), speed (parallel processing), and robustness to noise on inputs.

The adaptability of neural networks can cause degradation of the network if spurious disturbances are presented (Haykin, 1994, p 5). For success they therefore need to be both trained and operated on a specific stationary domain (though there are methods for non-stationary networks). Expanding a neural network, eg by adding new neurons, requires retraining which can be costly in terms of human expert involvement. As the size of the network increases so it becomes computationally more demanding, and the training time increases exponentially (Haykin, 1994, p 66). For effectiveness the network architecture should be structured to the problem at

hand, otherwise excessive training may be required. The structure represents the known context of the problem. Therefore neural networks are not entirely rule free. It is often difficult to understand what is happening, specifically how knowledge is represented, inside a neural network. The cumulative effect of the above characteristics is that neural networks can be difficult to scale up or redeploy to other domains.

The characteristic of not requiring explicit rules to be defined is an attractive one, (cf expert systems), and therefore neural networks have been used for a number of problems at early design. Noguchi (1998) applied neural networks to generate concepts for industrial design in a restricted case-study domain. Zhang and Fu (1998) used a neural network to predict packaging cost. Ivezic and Garrett (1998) developed a simulation-based decision support system (SB-DSS) for assisting early collaborative design. The neural networks described in the literature have all been developed on focussed domains. This is understandable, as doing so contains the problem size. No generic neural network exists that can undertake a variety of design problems, and it is difficult to see how one could be developed with existing technology given that it would require a huge case base to validate.

There is also a philosophical constraint on neural networks, in that they require the training cases to be consistent. Conversely, their simulation power is degraded if they are trained on test cases where the same set of inputs has in the past caused different outputs. Unfortunately these conditions frequently arise in design, as there are often many solutions to a given design problem. Neural networks also do not easily accommodate uncertainty in the inputs. Therefore, though neural networks are attractive because of their lack of formally defined rules and their ability to learn, they are likely to be successful in tightly focussed domains rather than trying to provide a universal solution to the early design stages.

2.1.7.7 Artificial intelligence and uncertainty

With the possible exception of expert systems, artificial intelligence does not process uncertainty very well. Many expert systems (but not all) are capable of dealing with uncertainty in a limited way. They do this by accommodating a *certainty factor* with the data supplied by the user, typically in the range from -1 to +1. In statistical terms this certainty factor would be equivalent to a confidence level. However it has little theoretical justification and is relatively limited in scope (does not extend into further statistical analysis). For example it would be surprising to find an expert system shell that would be able to accommodate data with a mean and standard deviation. However certainty factors are intuitively easy to use, and do not require substantial statistical knowledge. Expert systems have been developed with uncertainty in the input data processed using *fuzzy sets*, and uncertainty of decision modelled with the *analytic hierarchy process* (Hanratty et al, 1992).

The early design stages have possibly seen the least development in terms of expert systems, perhaps due to the creative nature of the task and the difficulties that these pose for artificial intelligence. The systems that do exist are generally concerned with focussed tasks within well defined constraints, such as design of specific systems. Other engineering tasks such as scheduling and manufacturing are more amenable to expert systems. However the diagnosis applications are the most developed of all. These systems are used for diagnosing faults in diverse systems (mechanical, electronic, medicine, etc.). The reason is that fault diagnosis is an essentially logical process and therefore suitable to modelling with expert systems.

Expert systems have their limitations, and cannot reasonably be expected to perform all artificial intelligence tasks. They are unsuited where the underlying rules are subject to change, and they can generally only achieve what a human expert could do, not more. On top of this, the expert system technology needs to be correctly applied and knowledge analysed adequately.

If an artificial intelligence system were to be used in this thesis, it would be a forward chaining one, as the need is for predictive rather than diagnostic capability. Expert systems have an advantage over functional modelling systems in that they cope with conditional logic, and this may be useful to the extent to which an early design can be modelled as a set of conditional statements. However even conditional statements may be inadequate to describe some of the functional relationships at early design, especially those relationships that are subjective and uncertain.

2.1.8 **TRIZ**

TRIZ is the Russian acronym for theory of inventive problem solving (TIPS). It is a methodology for inventive design. It originated with Genrich Altshuller in Russia, who analysed a large number of patents and observed that the principles fell into well-defined solution categories (Zlotin and Zusman, 1999).[33] Altshuller developed the TRIZ methodology around the principle that most solutions could be derived from existing knowledge that might be outside the experience of the designer or the industry. His intent was to develop a method for creative design that was not dependent on psychological tools or past experience of the designer (Domb, 1999).

The TRIZ method introduces the term *ideality*, which is the ratio of the useful effects to the harmful effects (TechOptimiser, 1997). Useful effects are the primary functional purposes of the technical system. Harmful effects are undesired consequences such as cost, power usage, pollution. A central principle of TRIZ is the “Law of Increasing Ideality”, which means that technical systems are generally improved over time

---

[33] TRIZ refers to five levels (TechOptimiser, 1997):

- Level one - Routine design problem. Solved by well known methods without the need for inventiveness. (32%)
- Level two - Minor improvement to existing system. Solved by methods known within the industry, but involving some compromise. (45%)
- Level three - Fundamental improvement to existing system, with compromises resolved. Solved by methods unknown in the industry, but known outside. Contradictions resolved. (18%)
- Level four - New principle. Solved by methods from science. (4%)
- Level five - Scientific discovery or pioneering invention. (1%)

towards greater benefits and reduced harmful effects. In the process the trade-offs in the design are reduced.

The TRIZ method has used patents to identify about 40 standard technical characteristics (including weight, area, power...) that cause conflict (TechOptimiser, 1997; Rawlinson, 1998). Part of the method is to state the technical conflict in terms of the required principle and the undesirable secondary principle. The next part of the method is to seek analogous solutions that can be adapted to the problem at hand. The analogous solutions correspond to forty *inventive principles* (including segmentation, nesting, inversion, self-service...) also extracted from patents. To solve design *contradictions*, TRIZ incorporates a table which suggests which inventive principles could be used for given combinations of undesired secondary effect and feature to improve. The function of TRIZ is to optimise systems in terms of operating principles, and it does not provide the function of an exhaustive combinatorial solution generator. The TRIZ methodology has been implemented in several ways. It originated as a paper-based methodology, and is apparently still used as such. However it has also developed into several software systems. As it includes a large database of patents, and principles, it is easier to use in software format.[34]

Since the user initially defines the problem in terms of function, the software can retrieve a list of effects that provide that type of function. Alternatively the user provides the feature to be improved and the feature that contradicts that improvement, and the system responds with the principles that could solve the problem. In order to achieve this, the TRIZ system has to be provided with certain information from the user. In particular the user has to describe the conflicts in function that occur in the model.

When defining the project for example in TechOptimiser version 3.0 (TechOptimiser, 1997) the user selects the objectives for the system, and indicates whether that

[34]Various TRIZ advocates have developed separate systems, eg Innovation Workbench from Ideation International (Inc) (http://www.ideationtriz.com), and TechOptimiser (TO) from Invention Machine Corporation (http://www.invention-machine.com).

objective should be maximised or minimised. The user also provides an estimate of how important that objective is, on a scale from one to ten. Then the user enters the functional model. The model is made up of three types of element: components (physical devices), super-systems (external systems that the user cannot change), and products (an event or end result). The user also sets up the interactions between these elements. The actions may be either useful (improves the capability of the system), or harmful (worsens capabilities). The model is made up as a graph (i.e. a network not a chart), by dragging and dropping the elements and actions. A link describes the relationship between any two elements. The system accommodates both qualitative and quantitative relationships. Both relationships are described by two values: the actual and the required value of the action. For qualitative relationships the description is by means of a slider, and for quantitative relationships it is with a numerical values and a tolerance. A significance factor is also provided, which is a measure of how important for project success it is for a value to reach the specified level. The user ranks each element according to function, problem contribution, and cost. There is provision for more than one team member to rank the elements, and the system averages the results. An element is *trimmed* if its action can be performed by another element (component or super system), or the component receiving the action can perform the action. The system also includes an "Effects module" which gives the user a database of engineering and scientific phenomena and effects.[35]

Comments on TRIZ

Sales literature exists for TRIZ, but there few impartial applications in the literature and it has thus been impractical to objectively assess the validity of all the claims made for TRIZ and its related systems. However the following are apparent:

- TRIZ specifically addresses the issues of design contradictions, also called *trade-offs*. This is an aspect of the distributed design problem identified by

[35]There are also extensions of TRIZ to other areas including:

- Anticipatory Failure Determination (AFD): a system that anticipates failure mechanisms, from Ideation International (Inc).
- TRIZ and QFD. Various claims are made in sales literature that TRIZ can compliment QFD, eg that TRIZ can solve contradictions in the roof of the House of Quality, and identify new functions to excite customers.

conventional design methodologies. However not all aspects of distributed design are necessarily addressed by TRIZ, such as the need to optimise components as a group.

- The TRIZ methodology addresses function and performance issues. It is primarily a system to identify possible solution principles. It does not address evaluation or assessment issues in any great depth. In the standard case it does not deal with failure modes, nor of robustness or reliability of design. It does not deal with probability distributions. It does not model viewpoints other than function.
- TRIZ specifically refers to the concept of a *technical system*, and uses it in the same sense as classicists such as Hubka and Eder (1988, 1996).
- TRIZ uses graphs to depict functional relationships. While some of the examples used in TRIZ literature specifically model power flow, TRIZ does not impose a bond graph[36] methodology or conservation of energy.

The TRIZ methodology requires that the user state the technical constraints explicitly, and there is no uncertainty permitted in the structure of the functional model. The requirement for explicit problem definition is common to many other solution generation methodologies eg expert systems, where it tends to limit the effectiveness of the methodology to less complex design cases. Most solution methodologies struggle with design problems requiring extensive innovation, as such cases tend to be difficult to decompose or to describe explicitly. Analogy suggests that TRIZ might likewise find operation difficult under such conditions, though this is speculation. There are too few documented applications of TRIZ in the refereed public literature to be able to judge fairly.

To clarify without suggesting criticism: TRIZ is not an uncertainty modelling tool, nor does it cover multiple viewpoints. Instead it focuses on solution generation for the physical domain, at which it excels. Hence it has gained its greatest popularity in engineering product and machine design rather than other design disciplines.

---

[36] The bonds in a bond graph are the connections and indicate the power flow between the various subsystems (Cellier, 2001).

2.1.9 Comments on inventive mechanisms

Generating alternative solutions is an important research topic in the design literature. Many methods have been proposed to assist or even partly automate the process. Perhaps the most common approach has been using morphological analysis in some form or another (manual or software assisted) to generate alternative solutions. Frequently artificial intelligence tools have been used to assist and partially automate this process. Such methods have been demonstrated to work, but they require a well defined domain and the selection criteria (constraints), have to be identified and programmed into the system. These are significant limitations and the results have often been disappointing. Fundamentally these systems operate with a limited knowledge base (solution fragments), and must try to find a solution to the problem in terms of these components. Further development is necessary before they can match a human expert for depth of reasoning. Artificial intelligence systems have performed best in well defined domains, where the tasks are relatively straightforward, routine or similar to previous tasks, but which require the processing of large amounts of data which a human finds tedious or forgets to do. Early design does not necessarily provide these conditions. Given existing tools it is therefore unlikely that artificial solution generation mechanisms will be more successful than a human designer except in niche areas.

The underlying premise of solution generation mechanisms, whether manual or assisted, is the belief that design will be enhanced if the solution space can be searched more exclusively, that designers sometimes ignore promising solution concepts through ignorance or prejudice. However it can be argued that it is more important to optimise the chance of success of the product than to optimise the inventive content of a solution. Expressed differently, the degree of inventiveness in a design solution is not necessarily positively correlated with the success of the product. Innovative products may win market share if the innovative feature differentiates the product from others in the market, but innovative products also struggle with issues of reliability, consistent product behaviour and quality, due to the

uncertainties that the new design introduces to manufacturing. An established design solution, even if unremarkable for novelty, could be the best solution if it robustly provides the required function. The inventive mechanisms are poor at considering (i) the robustness of the solution and (ii) aspects of the solution from other viewpoints. Some of the mechanisms, such as genetic algorithms, use fitness functions or other measures of utility. However these are quantitative and often arbitrary or at least subjective, whereas the qualitative nature of the requirements at early design makes it very difficult to formulate the functions in a defendable way, let alone extend that formulation to other aspects of robustness and other viewpoints. Other solution generation mechanisms, such as the manual processes, do not provide any solution assessment at all, leaving that to other processes.

2.1.9.1 **Distributed design**

The mechanisms for solution generation depend on functional *decomposition*, requiring that the problem be decomposed into smaller sub problems. The mechanisms then seek to find solutions to these sub-problems. For this to be successful it is essential that the sub-problems are sufficiently independent of each other that the solution principle used for one does not disturb another. Methods such as catalogue process, systematic idea generation, morphological analysis, case based reasoning, and TRIZ are subject to this requirement. In practice the subproblems are seldom completely independent (Finger and Dixon, 1989 a), and this is termed *distributed design* or *ill-structured design*. In these problems the solution decision for one part of the problem affects that for another part. Distributed design problems may be intractable with the above approaches, or compromised in terms of efficiency.

The dependencies are caused by:

- Components have to be optimised as a group, and therefore share space, cost, mass, and other resources.

- Components react with each other, eg vibration.
- Decisions in one part of the solution space affect other areas, especially service areas, (eg selecting a particular prime mover requires a specific power source elsewhere in the design).
- Conflicting requirements and constraints exist at higher levels (eg. a design may require low cost together with high stiffness).

Distributed design problems cause conflicting constraints: some constraint has to give at the expense of another. In the case of distributed design the selection of physical devices in mapping processes is conditional on the satisfaction of other possibly conflicting constraints.

To force functional independence on a problem when it does not really exist, for the purposes of fitting into a design tool, is to make sweeping simplifications that manifest as trivial solutions.

Distributed design problems occur frequently and are routinely solved by human designers. Human skills of conflict resolution are able to resolve these issues, but the methods require an element of judgement. Humans are good at solving distributed design problems, at least in the sense of finding a passable (even if sub-optimal) solution and convincing other humans of its merit. The automatic methods do not enjoy such successes. Other approaches to distributed design include:

- Constraint relaxation: Distributed design problems cause conflicting constraints, and some constraint has to give at the expense of another.
- Distributed design problems have been modelled in artificial intelligence using multiple problem solving agents (Finger and Dixon, 1989 a). These correspond to the *blackboard* concept in artificial intelligence. There are also artificial intelligence developments towards improving understanding of conflict resolution, and team interactions in design.

- Where parts of the design function are strongly inter-dependent, then the design process seeks to find a solution for the whole unit, using methods such as brainstorming.
- Experience and CBR.
- Some authors have addressed the distributed design problem with a morphological analysis approach, combinatorial synthesis of elements from a library. The implied hope is that if the library is sufficiently big, then an exhaustive search of the solution space will have been accomplished. There is still much work to be done before this approach can be considered successful let alone robust.

2.1.9.2 **Need for quantitative information**

Design tools are most mature at the parametric stage, with commercial CAD software having implemented the methodologies. At the configuration design stage there exist methods for automated selection of parameters based on optimisation methods (Finger and Dixon, 1989a). These optimisation methods require well defined, quantitative information. The lack of such information makes it difficult to apply the methods to the concept design stages. A limitation of many systematic processes is that they require quantitative measures or weights. This is the case for grammars, genetic algorithms, and simulated annealing. These quantitative relationships simply are not there at early design stages, and this limits the effectiveness of these methods.

2.2 Inventive constraints

In the case of incremental design it is likely that a design concept already exists and can be built on. With innovative design it is less likely that a prior concept exists, and in which case the inventive mechanisms and inventive constraints become critical to success. Origins of these constraints are discussed in this section.

2.2.1 Problem definition

Problem definition is that stage where the technical requirements of the design are distilled from the broader expression of need. Design problems are ill structured and under constrained, as they do not contain all the criteria by which to identify an acceptable solution. Therefore an important part of design is developing a complete and consistent set of requirements (Tang, 1996). As such the process involves an initial identification of need, which may be precipitated by external input. This is shown in Figure 2.5. The external input may originate from various other functions within the organisation, such as marketing, production, design itself, and indeed other sources. User (customer) feedback on a precursor product is also a potent stimulus to initiating the design process.

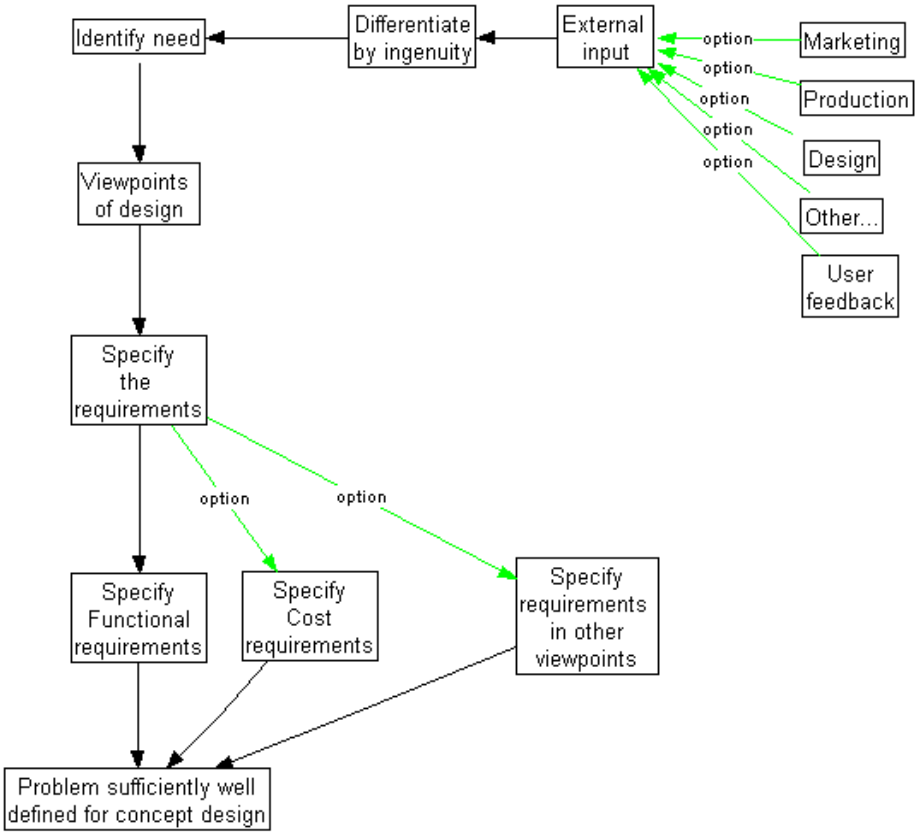


Figure 2.5: Problem definition

Having identified the need, the process aims to define the problem sufficiently well that concept design may proceed. Some of the possible building blocks along the way are shown in the figure, not that these should be seen as prescriptive. To define the problem sufficiently well usually requires that the problem be addressed from multiple viewpoints. These viewpoints include consideration of the technical function of the machine, along with issues of manufacturability, reliability, cost, ergonomics (ease of use), marketing, and potentially many more. Of all of these it is the first, the technical viewpoint, that is invariably well represented in engineering design, and possibly even over represented. Other viewpoints need to be considered, and one of the primary objectives of *concurrent engineering* is to provide these alternative perspectives sufficiently early in the design process that they can have adequate attention.

2.2.2 Focus on the customer

The fundamental principle of the quality philosophy is that the “quality of a product is its ability to satisfy the needs and expectations of the customers” (Bergman and Klefsjö, 1994). Earlier quality philosophy was on inspecting products for conformance to specifications, but the thinking has now shifted to building the quality into a product by focusing on customers. In the quality context a *customer* can be a person inside as well as outside the organisation. The term *product* is used in the broad sense to refer to a physical item and also to a service.

Customer needs and expectations are represented in the *Kano model*, which views customer satisfaction at three levels (Gustafsson, 1996). The *basic requirements* are those that the customer expects by default, and which will cause extreme dissatisfaction if they are not met. At the next level are *performance requirements*, which are features that customers will explicitly want in a product. At the highest level are *excitement requirements.* These are not anticipated or required by the customer, but none the less delight the customer.

The requirements of the customer are usually collected as part of a marketing process, and frequently referred to as the *voice of the customer*. There are a number of techniques used for this, and these have been formalised into what are called the *seven product planning tools* (*7 PP Tools)*. Briefly, these are described as follows (Gustafsson, 1996):

- *Group interview,* also called *focus groups*, which identify needs from a small group of customers.
- *Questionnaire survey* (or quantitative survey) to verify the needs from the group interview.
- *Positioning analysis*, a comparison of the product against those of competitors, a type of *benchmarking* with a focus on how the customer perceives the products.

- *Concept checklist*, which is a lateral thinking process that uses a checklist to explore alternative uses and configurations of a given concept.
- *Table-type conceptualising* combines results from various respondents into table format.
- *Conjoint analysis* is used to evaluate the preferences of customers and determine the attractiveness of various parts of the package they are offered.
- *Quality tables,* also called QFD.

Once the customer's requirements (possibly vague) have been found, there still exists the task of converting these into quantitative engineering specifications. This is where techniques such as quality function deployment (*QFD*), conjoint analysis and analytical hierarchy process have been used, but it would be naive to expect them to be perfect. The engineering specification represents the mapping of those customer needs into the engineering domain, and that mapping process is not necessarily either accurate or complete.

2.2.3 **Quality function deployment**

Quality function deployment (QFD) is a systematic method to transfer the *voice of the customer* (customer wants and needs) into engineering parameters that can be understood by the company (Gustafsson, 1996). It distills product attributes out of loosely defined customer requirements. The process is therefore analogous to the conversion of qualitative data to quantitative.

Bergman & Klefsjö (1994) list the necessary activities as:

(1) *Market analysis:* to explore customer expectations and needs.

(2) *Competition analysis*: find out their ability to satisfy customers.

(3) *Identify key factors*: these are the factors necessary for success of the product.

(4) *Translate*: Convert key factors into engineering characteristics suitable for design and production.

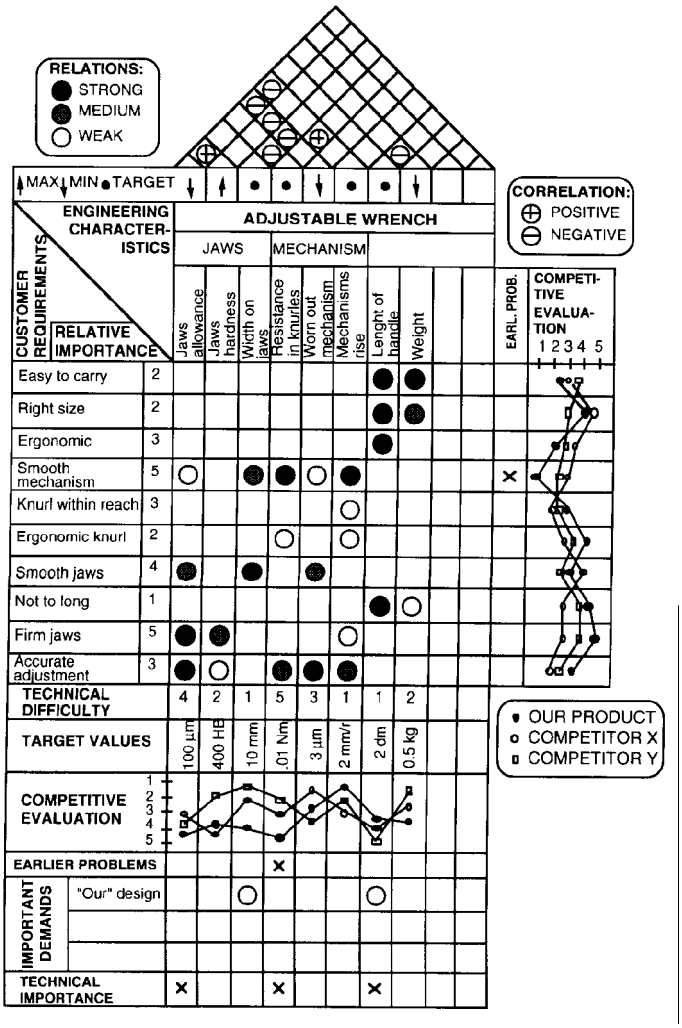


Figure 2.6: Quality Function Deployment uses a matrix to show the relationships between customer requirements (rows) and engineering characteristics (columns). The triangular matrix at the top is used to correlate engineering characteristics with each other. Other parts of the matrix show analysis of competitors products, and target values for the product. From Roland Andersson in Bergman & Klefsjö (1994).

Matrix

The process uses a matrix method to translate *Customer requirements* into *Engineering characteristics* of the product. Rows list the customer requirements, and the customer's evaluation of the current product and competitive products is also shown. Columns contain the engineering characteristics, or design attributes. These do not necessarily imply solutions. Symbols show the relations between the customer requirements and the engineering characteristics, see Figure 2.6. A triangular correlation matrix can also be added above the engineering characteristics, to show how the engineering characteristics affect each other. This shows whether an improvement in one parameter will cause an improvement or

worsening in another. The resulting roof shape gives rise to one of the alternative names for the method: *House of Quality*.

At the end of this stage the critical engineering characteristics are identified and prioritised. These characteristics become the criteria for the new product. A benchmarking can be included, to show how the current and the competitive products perform against these characteristics.

Four phase model

The QFD process may be extending into other processes within product development, as shown in Figure 2.7. The first stage is to translate *Customer requirements* into *Engineering characteristics* of the product. The latter are then transferred to another matrix, where they are transformed into *Part characteristics*. In turn the part characteristics are translated into *Key process operations*, and these in turn into *Production requirements*. In this way the method seeks to transform customer requirements into production details. According to Gustafsson (1996) the four phase QFD is more a “conceptual approach than a concrete description”. Gustafsson also points out that the process does not support the design phase of a product but jumps over it. The strongest and perhaps most useful area of QFD may be in the initial conversion of customer requirements to engineering characteristics.

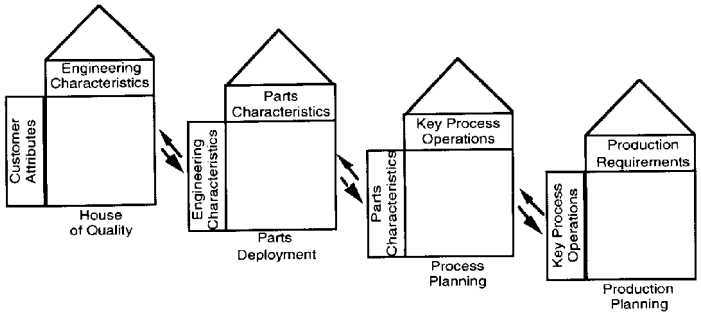


Figure 2.7: QFD aims to progressively develop the product from customer requirements through to Production requirements. The output from one matrix is fed in as the input to the mext. From Bergman & Klefsjö (1994).

Comments on QFD

One of the marked differences between QFD and conventional engineering design approaches is the emphasis on customer requirements. These feature in the

conventional design processes, but mostly as well defined engineering specifications. QFD actively seeks to convert ill-defined customer expectations into firmer engineering specifications. However it is a method that involves a certain amount of subjectivity, and this can introduce unreliability into the results.

For QFD to be successful, it needs to be applied by a multi-disciplinary team, and this presupposes the existence of the necessary corporate culture with investment in training, facilitation and market research (Gustafsson, 1996). These costs can be more significant than conventional development processes, although there is the general expectation that the benefits are also significant.

The scoring system in QFD uses the *value analysis* method. Each of the customer requirements is listed and given a weight to indicate its importance. The weight is usually out of ten. Each of the engineering characteristics is then studied and scored. The score is based on how much that engineering characteristic contributes to the Customer requirement. After that, a Total raw score may be calculated for each engineering characteristic. This Total raw score takes the products of each score and the customer weight, and sums them. Finally the Total raw scores can be normalised, giving a relative weight for each engineering characteristic.

At the next phase engineering characteristics are to be converted into product characteristics. The engineering characteristics become the input rows using the relative weights found previously. The physical components in the system become the columns, and the process repeats. At the end of the second phase the relative weights of the parts will have been established.

The QFD method assigns a numerical score to the relative importance of the customer requirements. This score represents the opinion of those setting up the analysis and is therefore subjective and belief based. Perhaps more significantly, the scoring system imposes a numerical ranking on the customer requirements. These scores are then propagated through the matrix to determine the engineering characteristics of the product. Inaccuracies in the initial scoring can have a significant

effect on the predicted outcomes. Moreover, the QFD method has no means to accommodate uncertainty in either the scoring or the relationships. Mill (1994) discusses limitations of the QFD process, particularly "the difficulty of listening to the voice, misinterpretation of message, constantly changing customer needs and too many customers and needs".

An interesting feature of QFD is the way it accommodates aspects of distributed design. It does so through the triangular correlation matrix at the top of the diagram. The relationships are expressed in terms of positive, negative, or no correlation. While subjective, there is at least some means to explicitly state the distributed nature of the design at the early concept stages.

2.2.4 **Conjoint analysis**

A vital part of implementing the quality approach is the *market analysis* that seeks to identify the product characteristics that are attractive to customers. This is usually complicated by the grouping of several characteristics in one product. A customer's preference may depend on the mix of various characteristics, or *bundles of attributes.*

Conjoint analysis is the study of the joint effects of bundled attributes. It is a method to estimate the influence of each separate attribute, even though the attributes are not presented singly. In use a variety of concepts are put together, each with different attributes. These concepts are presented to customers for evaluation. At the simplest level the evaluation may be done with *paired comparisons*, and the customer responds only by selecting the preferred concept. In the *analytical hierarchy process (AHP),* respondents select the weight in paired comparisons. This takes into account how much better the one concept is than the other. The limitation can be the large number of comparisons to be made, and to alleviate this the decision tree is divided hierarchically into smaller portions. Another approach to conjoint analysis is the *full profile approach*. This uses concepts (bundles of attributes) that are carefully

selected so that the effect of one attribute is independent of another. This is an application of a fractional *designed experiment*. By using this approach it is unnecessary to ask the respondent to evaluate every possible combination of attributes, instead a smaller number of concepts can be presented and the value or utility of the attributes can still be estimated. The limitation of the full profile approach is that the respondent has to evaluate complete concepts, which can be difficult if there are numerous concepts (Gustafsson, 1996).

The conjoint analysis method is typically used to determine customer preferences for product features. As such it is applied relatively late in the product development process, when features are firm enough to present. The AHP method is commonly used to rate the customer needs for QFD, and full profile conjoint analysis has also been applied to this task by Gustafsson (1996), who comment that conjoint analysis assumes that “products are decomposable into separate attributes”. The conjoint method is also applied to market analysis and pricing issues. Calantone et al (1999) discuss the importance of screening new product ideas, and they use AHP for this. Perego and Rangone (1996) note that though the AHP can address trade-offs between quantitative and qualitative measures, there are difficulties with using quantitative measures expressed in a standard scale, and composing quantitative and qualitative information in an unbiased manner.

Other perspectives

Another view of uncertain specifications is provided by Antonsson & Otto (1995) who use the term *imprecision* to refer to ‘vagueness of a preliminary, incomplete design specification’ or ‘uncertainty in choosing among alternative’. Imprecision in their usage corresponds to the fuzziness of *fuzzy theory*, and indeed this is the method that they use to quantify imprecision. Their methods are described more fully later.

2.3 **Simulation and other Mechanisms to assess solutions**

Design assessment is closely linked to modelling, in that the results of any simulation (whether functional modelling, artificial intelligence, or decision analysis process) need to be assessed in some way. Assessing a proposed solution requires the use of assessment mechanisms and the availability of assessment constraints. The constraints may either arise from concurrent engineering requirements elsewhere in the project, or be of endogenous professional judgement. This section reviews the literature on assessment mechanisms. Primarily these are methods and tools for modelling the performance (or function) of engineering systems. The thesis concerns modelling performance at early design and therefore the review has a particular focus on how others have accommodated uncertainty in their methodologies. While most of the discussion in this section is on simulation, it should be noted that testing (eg of physical prototypes) is another assessment method. For example Bach (1999) describes testing as a "process of developing an assessment of product quality", and notes importance of managing risk and quality. In simple designs there is not great difficulty in assessing the concepts and selecting one above another. Concepts may be evaluated by various methods, human decision making processes being the default method. However many if not most designs are not so easily assessed.

Santillan-Gutierrez and Wright (1996) view the conceptual process as (1) generate configurations, (2) evaluate them, and (3) select one configuration. They comment that it is a “continuous process of evaluation and satisfaction of different criteria, dealing with often vague and imprecise information”. It is sometimes difficult to model function, and constraints due to marketing and human factors are even harder to incorporate into formal assessment methods.

Law and Kelton (1991) distinguish between different types of simulation systems:

Discrete event simulation
This models systems where discrete events occur over time, such as the arrival of a customer in a queue. Most of Law and Kelton's applications are in this category, and involve the calculation of system performance (eg queueing times, inventory levels) where each of the inputs is simulated (eg with exponential random variables). Calculations are repeated with different configurations to seek optimum solutions.

Continuous simulation
This models system behaviour over time, eg transient behaviour. The output is one or more system parameters charted against time. To do this it uses a mathematical model of the system, typically differential equations. In some cases these equations may be solved exactly, otherwise it is necessary to use numerical solutions. The continuous simulations are entirely deterministic, in that there is no uncertainty. There are also models that combine both discrete event and continuous simulation.

Monte Carlo simulation
This method is applied to systems to determine the behaviour at one point in time. It is not primarily a dynamic analysis method. Monte Carlo uses random numbers to solve system problems that are not analytically solvable.

2.3.1 **Functional modelling**

The term *functional modelling* is widely used in the design literature, though the meaning varies with to the context. Some authors equate it to the process of developing sub-functions by a decomposition process, and the subsequent assignment of solution principles to those functions. In such a context the *model* refers to the reduction of a design problem to a set of functions. The assignment of

physical devices to meet those functions is also sometimes included. Another use for *functional modelling* is for the creation of a model of design intent. This type of model may use natural language models, being a word-based description of design intent. Yet other authors use the term *functional modelling* to refer to the creation of a model of some behaviour (typically energy flows), and the procurement of analysis results by simulation using this model. These are analytical models, and they typically define the system as a network of devices and the mathematical relationships between the devices. They simulate the behaviour of a system, for example the concept of *technical systems* from Hubka and Eder (1988, 1996).

In many ways there is a logical progression from functional decomposition, through to a qualitative modelling of design intent, and on to simulation and prediction of machine behaviour. Afterwards the sequence can be envisioned to continue to detailed design and other engineering analyses.

2.3.1.1 Modelling energy, materials and signals

Early work on functional modelling was done by the German design community (Pahl and Beitz, 1988), which developed a model of the proposed machine based on flows of *energy, materials and signals* between the parts of the machine. There are three separate models, one for each of energy, materials and signals. The idea is to consider any system as a set of black boxes connected by input and output flows. The models are based on principles well known to engineering, that of conservation of energy for the energy flow model, conservation of mass for the material flow model, and control system dynamics for the signal model. The historical basis is documented in Pahl and Beitz (1988), who refer to earlier works by Rodenacker, Roth, and Koller inter alia. The early developments in functional modelling used a grammar of standard terms to describe the models. For example Rodenacker's model as reported in Pahl and Beitz (1988) defines *separation*, *connection* and *channelling* (of flows of energy, material or signals). The functional relationships may

be defined at various levels of complexity. In particular the method deals with *quantity* and *quality* of the following functional flows:

- Energy: mechanical, thermal, electrical, chemical, optical, nuclear, etc.
- Material state (gas, liquid, solid, dust...), and other attributes(raw material, test sample, workpiece, end product...)
- Signals: magnitude, display, control impulse, data, information, etc.

The classical energy-material-signal method forms the basis for a systems thinking approach to design. The method aids comprehension of the issues involved in the design, and is therefore an analysis approach. However it is not a simulation system, as it does not contain its own simulation engine. Instead it relies on the designer to select and perform any analyses using exogenous tools. A novice designer could potentially apply the analysis and then not know what to do after that. It therefore relies on an expert designer[37]. In some cases the method has been developed into a design tool that encapsulates some degree of expert knowledge for the benefit of less experienced designers. Energy-only models are perhaps the most common application. A development of the concept is that of Deng et al (2000) who model flow of action, thereby modelling design intent rather than physical flow of energy/materials/signals.

2.3.1.2 **Simulation of system behaviour**

A significant part of current design research concerns the means to simulate some aspect of the behaviour of a machine system. Many studies have looked at the modelling of engineering design from the viewpoint of function, and indeed this is a fundamental part of conventional engineering analysis and design. The foundation for analysis of behaviour is a model of physical devices (such as gears, springs, actuators, bearings, slides, etc.). These devices are based on *qualitative physics.*

[37] It is possible that many expert designers approach design systematically anyway, and therefore would obtain little additional understanding of their system by creating an 'energy-materials-signals' model. Most of the known examples (eg Pahl and Beitz, 1988) are relatively minor design cases.

They are devices with known physical principles, though the actual values of the parameters (eg number of teeth for gears) may not have been assigned. Descriptions of physical devices may be stored in a library, and made available for the *catalogue* design process. Quantitative simulation of machine behaviour requires a model that includes the necessary quantitative model. Input-output transformations (mathematical relationships between parameters) are a common approach.

One of the methods that has been used is that of *bond graphs*. These are an application of graph theory, with energy flow as the viewpoint. *Nodes*, also called *vertices*, correspond to the physical devices. They are connected by *arcs*, also called *edges* or links, which are the connecting lines on the graph and represent the conduits for the energy flows between the devices. The models apply principles of conservation of energy to create block diagrams of the energy flows.[38] The "bond" refers to the bonding of devices (nodes) at the ports where energy flows in and out. Bond graphs require that only compatible links (in the energy sense) may be made. This feature may be used in the construction of the model in the first place, since appropriate devices may be selected from a library or catalogue of devices. Once the bond graph has been completed and the transformation relationships furnished, then the system may be simulated. An example of an implemented system is "Schembuilder", in which bond graphs are used to create a model of power flow and then simulate the transient behaviour of the system. The system is described in further detail below. Gui & Mantyla (1994) represent a design as a combination of three models, connected by graphs. The models are:

- functional model carries the specifications for required functions
- device model is of physical structures
- process model covers manufacturing.

[38]Bond graphs may use *across* (or force or effort) and *through* (or velocity or flow) variables, and their product being units of power (Cellier 2001, Broenink 2001). The across variables are measured at two points, and examples are voltage, pressure, temperature, displacement. The through variables are measured at one point, and include current, discharge, heat flux, force. The product of the across and through variables is power (or energy).

There exist various *natural language* approaches to functional modelling (eg Bracewell and Sharpe 1996; Deng et al 1998). In these approaches function is described with natural language fragments of the type *verb, noun, complement*. These are closely related to the grammar inventive methods described above.

An example of a natural language approach is that of Deng et al (1998) who model functional performance in terms of energy flows (storing, transmitting, converging, and changing), and using natural language descriptions. They feel that existing systems based on verb-noun plus qualifier are inadequate and propose the use of *object oriented* classes with multiple attributes. Their *class* (a template structure in object oriented programming) is defined in terms of the following attributes:

- Name, a verb noun pair, eg “transmit motion”. The verb may be store, transmit converge, branch, or change, and defines the action to be performed. The noun describes the target.
- Complement, a qualifier, eg “flexibly” or an adverb like “greatly”.
- Type, the function being considered. They address function from a performance viewpoint only, but others are provided for.
- Level at which the function operates: overall functions (most general and abstract), embodiment functions (decomposed from overall functions), and geometric functions (for specific geometric at detailed design).

Specific functions may then be defined as instances of these classes, in that they inherit the attributes of the class and may add other attributes such as the names of the input and output variables. The mathematical relationships between input and output variables may also be provided. Implementation is in an object oriented programming language. The *classes* are generic prototype physical structures (eg gears in general), and the *instance*s are embodiments (eg a particular type of gear such as helical gear). The system may be used to guide the user in searching for prototype structures, although the system does not check whether the retrieved function is appropriate. A *library* contains the fundamental functions and their associated design elements, so that physical devices may be mapped to the decomposed structure. In a subsequent paper Deng et al (1999) further describe the mapping of function (‘intended input action’) through a behaviour (‘intended output

action') to a physical component. The design is thereby represented as text statements.

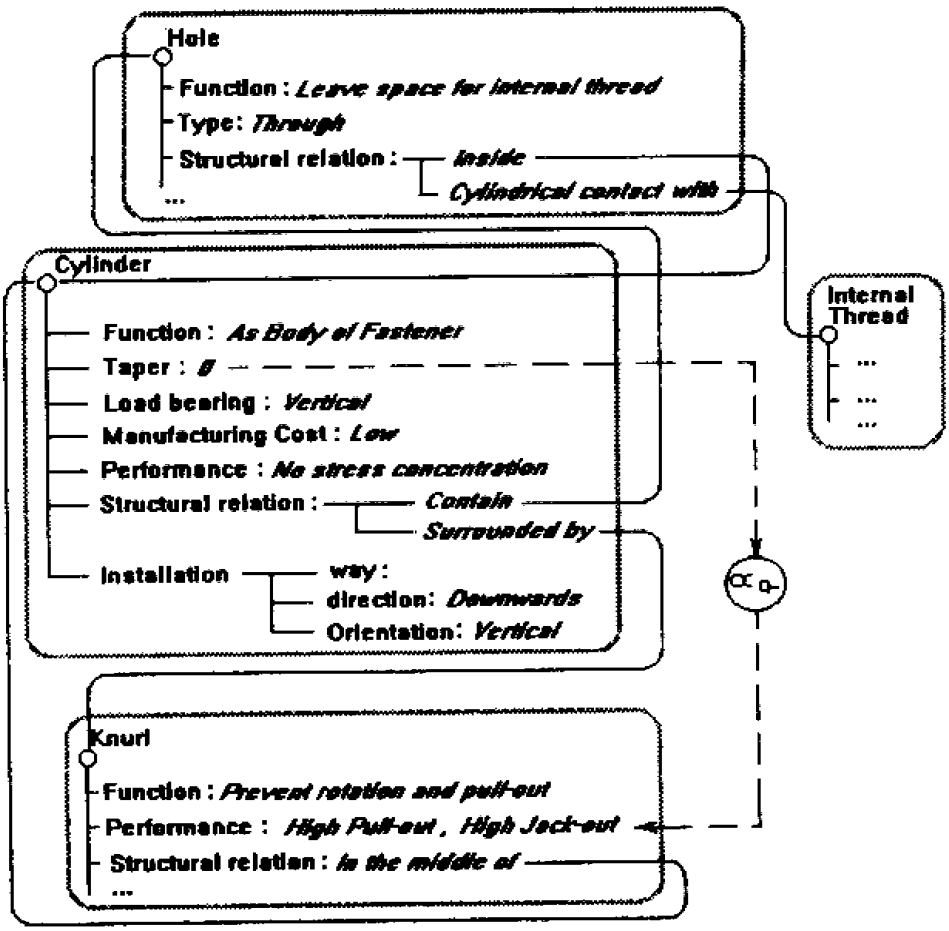


Figure 2.8: Nodal view from Zhong and Dooner (1996).

Zhong and Dooner (1996) use a *nodal view* of the configuration, which is a tree-structure of the component features and their relationships. The method shows structural relationships (solid lines) and causal relationships (dashed lines), see Figure 2.8. The system qualitatively evaluates mechanical fasteners.

Tang (1996) proposes developing an artificial intelligence system in which design components are given input and output functions, and the transformation between them. They also have attributes to describe physical and geometric features. Reasoning modules and software handlers are attached to the object classes, so that when a change is made to an instance of the object then the consequences are

determined. Rules are used for special purpose reasoning. An instance of a product model may be built up by the designer selecting components from the knowledge base, or by functional synthesis from the functional requirements. This model only describes the physical connections and the orientation of the components. Later in the embodiment design process the values of the individual parameters are determined. The expert systems monitor the changes that the designer makes to the product model, and infer as much as they are capable. The designer is primarily responsible for exploring the design.

Gui and Mantyla (1994) develop a model for *option design* whereby components are replaced by other components with the same function. The method has similarities to morphological analysis, except that it is applied to existing designs and not to create new concept variants. The system investigates the propagation of change through the design, identifying inconsistencies.

Bond graphs have been used in *functional decomposition* (Bracewell et al, 1996a,b). The method is based on *graph theory*, with the limitation that only compatible ports on nodes may be connected together.

Comment
The natural language functional modelling systems do not analyse dynamic behaviour (changing function with time), nor do they incorporate randomness (process variability) or uncertainty of analysis. They require a quantitative system, so they are unable to process uncertain relationships. They also require that the proposed design be able to be functionally decomposed, which is a requirement for independence of sub-functions.

Virtually all machines have different *states* of behaviour, or *modes* of operation, which change with time. For example a dishwashing machine goes through wash and drying cycles, where different sub-systems of the machine are switched on or off as necessary. These states of behaviour are discrete in the dishwasher example, since the machine switches into from one state into the next. However there are other

situations where a machine may have behaviour states that continuously vary with time. For example an electric motor has a transient behaviour in terms of speed and torque as it is started. States of transient behaviour exist whenever a machine contains storage elements, such as momentum (energy domain), tanks (material domain) or capacitance (signal domain). Implicit in the original model of energy-material-signal is the notion of time, at least in the signal domain. However most applications of the method have considered steady-state models. Discrete behaviour states do not appear to have been incorporated into models. Continuous states are accommodated in at least one model, that of “Schemebuilder”, which models transient energy flows.

Many systems have been proposed in the literature, and several have been built for specific domains. One of the more successful and developed simulation systems is described below.

2.3.1.3 **Case study: Schemebuilder**

“Schemebuilder” is a system for concept and embodiment design. It assists the development of a qualitative concept scheme into a quantitative embodiment. It models the design using principles of *conservation of energy*, and is implemented with *bond graphs*[39]. The system has been developed at Lancaster University Engineering Design Centre. Various aspects of the project are described in Oh and Sharpe (1996), Bracewell et al (1996a,b), Oh et al (1994).

The system provides decision support to the designer during the process of design. It helps the designer assess a design scheme from a functional viewpoint, and it does this principally by functional simulation. In this context a *scheme* is a design solution where the components have been selected so that the major functions are provided. The Schemebuilder system aims to provide the means to quickly evaluate alternative

[39]Bond graphs are graphs (cf graph theory rather than a chart) with nodes (blocks) connected by arcs (lines), with an underlying software mechanism that only permits appropriate arcs to be drawn (eg both ends must be ‘energy’).

schemes. The underlying philosophy of the authors is away from “automated design using the expert system approach”, and instead to “provide decision support and allow the human designer to apply the judgement” (Bracewell et al, 1996a).

Operation: In use the designer enters the desired function for the system or the block being considered, and searches for possible solution elements in a database. The system provides a library of working principles for this purpose. Past design schemes may also be used as the starting point for new designs. Alternatively the required function may be decomposed and the sub-functions investigated. Each scheme represents the minimum functionality of the design, at a qualitative level. Spatial arrangement of the design is not provided unless it is explicitly defined as a requirement. In building the model in Schemebuilder, the designer is creating a block diagram (graph) based on energy and data flows, and it is this model that the system subsequently analyses. Gross energy mismatches between components can be detected by the system. Schemebuilder may also be used for simulation in embodiment design, once the designer has provided sufficient detail about the components in the design. The designer selects assessment criteria (eg cost, weight, working temperature, maintenance interval) from a list for the component type, and the system provides a comparison using a bar chart.

The simulation engine in Schemebuilder is a commercially available block diagram simulation software called “*Simulink*” (Math Works Inc). Every component is modelled, using both effort and flow variables. Components may be grouped into higher level components, with a number of inputs and outputs. These are collected in a database. Components need to be defined at multiple levels of complexity, though this appears to be a constraint of the simulation

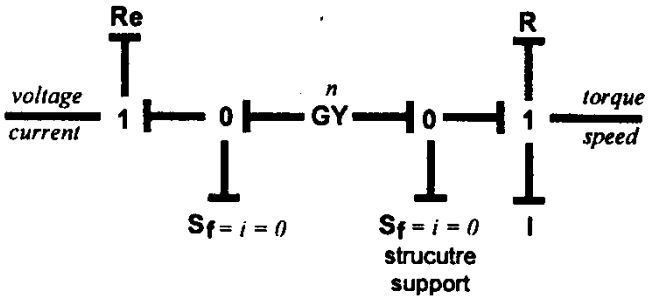


Figure 2.9: Bond graph model of DC servomotor at its second level of complexity, using Simulink, Bracewell at al (1989b).

software rather than a fundamental requirement. An example of a bond graph for a DC servomotor is shown in Figure 2.9. The whole system is then modelled as a control system with building blocks such as these. The simulation is then able to determine transient behaviour, and the example of the system response to a step change in voltage is given. Schemebuilder also incorporates a layout tool. This provides a CAD system (*AutoCAD*) with the outside envelope of the system components, and their locations with respect to each other. The spatial layout of the design may thus be visualised[40].

Architecture: Schemebuilder integrates multiple software systems. Though Schemebuilder uses rules, it does not appear to use expert systems, or any other artificial intelligence tool for that matter. Instead it uses databases and modelling techniques to simulate the functional behaviour of a design. It is essentially the integration of several systems.

Use of conservation of energy principles permits the system to be used across multiple domains. The functional modelling in Schemebuilder incorporates data and energy functions. A limited number of functions are available, including transformer, gyrator, resistor, compliant energy store, inertial energy store, common effort junction, common flow junction, source of flow, and source of effort. Each link (arc) is given a unique identification number, and components have ports to which the links are attached. A *relational database* is used to contain the functions and their links. The system permits single component types to appear at different places in the structure, where they embody different functions. Furthermore a component is also permitted to embody more than one function (if appropriate). Each block may be progressively developed to deeper levels. Background information and technical notes are provided by means of hypertext links to other notes. The Schembuilder system consists of several databases, including ‘Dictionary of working principles’,

[40]The drawings show only large scale features, and they give the example of a motor which is modelled by a cylinder for the body, another cylinder for the shaft, a plane for the mounting surface, an axis for the shaft, and a point where the part connects to the next part. The axes permit alignment of the parts to be done automatically.

‘Function means’, and ‘Embodiment component’. The system is held together by the ‘MetaCard’ hypermedia development tool.

Limitations of Schemebuilder are the need for well defined mathematical relationships if the simulation is to be effective. The system records design intent, but does not process it directly. Schemebuilder goes a long way towards addressing issues of providing tools to assist the early concept design stages. Its creators have been conscious of the need to avoid hindering the creativity of the designer and they give the designer control over the design process. They have managed to successfully integrate a variety of tools into one user environment.

2.3.1.4 **Feature based modelling**

The feature based modelling approaches are based on *geometric* features. *Features* are the critical areas in a design, typically the parts of the geometry that affect function. Geometric shape determines much of what is important in mechanical systems, like function, aesthetics, manufacturability (Fu and de Pennington, 1994). However the term is sometimes used more widely than just geometry, extending to any part of the design that is important in reasoning (Finger & Dixon, 1989b).

Engineering parts have critical features that are important for function, linked by non-critical surfaces. Geometric (eg CAD) models do not differentiate between the features. Artificial intelligence systems are overloaded by the excess detail, necessitating extraction of the critical features from existing geometric models (Finger & Dixon, 1989b).

Feature based models might appear to be relevant only to the detailed designs stages where the necessary geometric definition exists. Systems that design-with-features are based on manufacturing processes (Finger & Dixon, 1989b). Fu and de Pennington (1994) evaluate product functionality and performance, using *geometric*

reasoning based on graph grammar parsing. Features in a solid model are represented by a *graph* grammar. In this graph method a node represents a surface, and a link represents a dimension. The *grammar* describing the component may then be processed and manipulated (parsed) to extract geometric reasoning. The parsing process infers features that may not have been obvious in the original graph. The intent of this type of research is directed at feature recognition, and to relate this to reasoning on design function or manufacturing characteristics.

Gaun and MacCallum (1996) support the maintenance of a principle of *minimum commitment modelling* so that CAD systems for early stages of design should not force the designer into unnecessarily early commitments or decisions. They apply this to geometric design where such commitments include arrangement, shape, size and position of components. They argue that systems need to be able to support both vague and precise geometric information, and that the user interface should not hinder the designer's interaction with the system. They point out that most CAD systems require precise information that is unavailable in the early design stages. The issue of positioning components in a geometric space is addressed, using the notion of *uncertain region* for a component. This models the uncertainty of location of a component. They raise the issue of imprecise size, though it is not addressed in the study. Their work involves the development of a system for geometric positioning, using *Lisp* object oriented language (CLOS), a constraint solver (CLP(R)), and a solid modeller (ACIS). Particular features are:

- imprecise positioning of components is possible
- later addition of refinement and more constraints is possible
- solving location constraints
- propagating results through the model
- a new configuration is checked for inconsistencies.

Brady and Juster (1996) have worked on the manipulation of incomplete *geometry* during concept design. Their viewpoint is from *assembly*, and the geometry features that effect assembly. The fundamental building blocks are areas on components, since these interface with each other, see Figure 2.10. These are partial

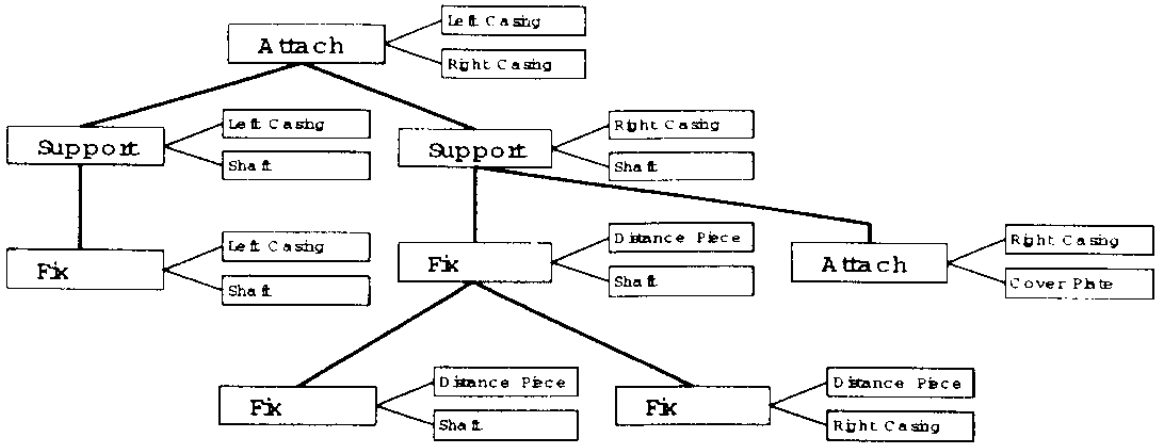


Figure 2.10: Function structure from Brady & Juster (1996)

components, and they are stored in an object oriented *library*. Their system, called CDT (*conceptual design technique*), takes a function structure that has been specified by the designer, and searches the library for solution principles. The results are presented to the designer, as combinations of solutions. Depending on the number of components, a potentially large number of solutions may be generated. However the system does not comment on the suitability of individual solutions. The Library is based on object oriented partial-components, which are described in three related data structures (or *planes*), namely:

- *Description*, using a taxonomy of form features eg threads, flanges.
- *Function* within the assembly. Available terms are *attach, fix, locate, seal,* and *support.* They have classified assemblies into two classes: Assembly functions (which hold the assembly together and do not affect dynamic properties), and Product functions (which describe the functional requirements of the completed product).
- *Geometry*.

The library may be expanded by adding new partial components.

2.3.1.5 Functional modelling and uncertainties at early design

The functional modelling systems are strongest at detailed design, since they operate most easily on quantitative relationships. They have limited ability to process uncertainty, and this would tend to suggest that they would be deployed with difficulty at early design. Indeed, while some functional modelling systems have been extended into the early design stages, they remain tightly focussed on a single problem space and could require major rework to be used in other applications.

Feature based models appear to have more potential for the early design stages, as some of the systems have been demonstrated to be able to operate with incomplete geometry. However Finger & Dixon (1989b) express uncertainty whether a feature based representation will be able to be interpreted from different points of view such as function, assembly and cost. Indeed, the feature based models primarily explore geometry viewpoints (typically size and relative location/fixation of parts) and there is no immediately apparent manner in which some of the non-geometry viewpoints could be represented or simulated.

2.3.2 Assessing uncertainty and variability

There are several well established tools for assessing the variability in a simulation result. *Sensitivity analysis* seeks to determine how much an input parameter affects the output. One-way sensitivity analysis [41] shows which variables on their own are critical to the outcome, and would benefit from closer scrutiny. However the method

[41] One-way sensitivity analysis is the simplest method, as it involves taking each factor in turn and varying it between the expected lower and upper limits, or three values (eg minimum, expected, and maximum). The output (eg expected value) from the decision model is determined at each step. The effect of changing the factor is then evident on the output. Each factor may then be varied in turn, independently of the others, and the effects observed in the same way. The results may be shown in a *tornado diagram*. In this diagram each factor is assigned a bar denoting the range in output for which it is responsible. These bars are stacked with the most sensitive variable (longest bar) on the top.

does not show the effect of one variable being high while another one is low, let along the interaction of multiple variables. Also, the probability of an outcome is not determined. Two-way sensitivity analysis [42] overcomes some of the limitations of the one-way method, but it is limited to pairs of variable at any one time. The method can be adapted to explore the sensitivity to probabilities, and extended to multiple alternatives (not the same as multiple variables though). This is not common, perhaps because the results become more difficult to interpret. *Scenario analysis* is similar to Sensitivity analysis except that it identifies which scenarios (combinations of input parameters) lead to a given output value. At a higher level of complexity it is possible to model the input parameter as a complete probability distribution and then determine the output, though more powerful computational methods such as *Monte Carlo* simulation are then necessary.

Designed experiments

The experiments are fractional factorial designs that seek to identify which parameters are most important for product success[43]. However when there are many factors involved in determining product performance, then the design of experiments becomes cumbersome. It is possible to do the analysis and experimentation in phases, using a broad brush at first and refining the parameters of interest in subsequent experiments. Closely related topics are *conjoint analysis* and *analytical hierarchy process* (discussed above).

[42]Two-way sensitivity analysis is an extension of the one-way sensitivity analysis. It takes two variables, runs them through their likely ranges, and determines the outcome for each combination. The results are plotted in the x and y space, and for a given output criterion there will be pass and fail regions separated by a straight line. The nominal values of the variables may be plotted too, and the closeness to the dividing line shows how critical the variable is to the outcome.

[43]The experiments systematically vary a number of candidate parameters, and from the results deductions are made of the likely critical parameters. These are parameters that have the most effect on product performance and are the ones to watch most closely in production, particularly their process variation. Furthermore, robustness may be designed into a product by finding values of the parameters that minimise the sensitivity to variation (the *dispersion*).

Loss functions

Loss functions are a quality contribution to design. They seek to quantify the deviation from the target value for the parameter of interest[44]. Although the method is concerned with process variability, it is a design approach rather than a method of process quality control. The *Taguchi* approach uses the mean square deviation. More conventional definitions of loss only consider loss to occur when the process variable strays outside the tolerance band set by the designer, in other words that all values between the upper and lower specification are equally desirable. This is not particularly reasonable, so using the mean square of the deviation from the target has some attraction. There remains however the difficulty of quantifying real economic losses for some parameters. A critique of Taguchi's method is provided by Box et al (1988).

2.3.3 **Fuzzy theory**

Fuzzy theory (or *fuzzy logic*) provides another methodology to simulate quantitative systems. It can also accommodate qualitative parameters providing they can be placed on a numerical scale. Fuzzy theory relates not so much to probability as to possibility. It provides for a sliding scale between two extremes, so that an answer can be permitted to be more than just *yes* or *no*, but rather a spectrum of responses between the two extremes. An object may have any grade of membership of the range, between complete membership (1) and non-membership (0). [45]

---

[44]The objective is to minimise the quality loss over the life cycle of the product. Or putting it the other way, to maximise the robustness of the design to manufacturing and operating deviations.

[45]A Fuzzy number consists of a pair (x,u) where x is the element (eg length) and u is the degree of membership, eg (1.83, 0.6). A fuzzy set consists of a set of such fuzzy numbers { (x1,u1), (x2,u2), ..., (xn,un) } (Stachowicz and Beall, 2001). In this way the x dimension can take on a numerical range x1 to xn to model a physical parameter such as length, and the u membership can be represented by a function that looks like a probability density. Many fuzzy applications use a triangle membership function for convenience and ease of processing, though other functions are possible. An attractive feature of fuzzy sets is that they are not limited to numerical ranges, i.e. the x dimension can take discrete textual values such as 'low' or 'high' etc.

Quelch and Cameron (1994) motivate for the use of fuzzy sets, on the grounds of its non-statistical nature and the reduced computation effort. They suggest that fuzzy set theory might also model uncertainty and expert opinion. Bazu (1995) used a lognormal distribution to model reliability in a manufacturing domain using fuzzy theory. Quin and Widera (1996) used fuzzy sets in failure modes, effects, criticality analysis (FMECA). Fuzzy sets have been applied to *expert systems*, where the membership of the inputs is used to determine to what degree the output defined by the rule applies. Fuzzy theory thereby supplements the inference capabilities in a similar way to confidence factors. However the mathematical basis for fuzzy logic is better than for confidence factors. A limited number of logic operators are used in expert system applications.

Fuzzy theory has been applied to the *early design* stages. Notable developments include the *Method of Imprecision* (MoI) developed by Wood & Antonsson (1989), Antonsson & Otto (1995). They use fuzzy sets to describe approximate design parameters that are numerical in nature. They provide an example of a stress calculation using variable lengths and forces as inputs. They re-interpret the fuzzy membership function as a set of preference values rather than vagueness in meaning, i.e. indicating the preference of the designer for various settings of a parameter. They use α-level cuts to compute fuzzy arithmetic outputs. Monte Carlo simulation would have produced similar and even more accurate results (since fuzzy set arithmetic imposes a loss of information) but that is not to detract from their accomplishments. As with other arithmetic modelling systems operating on numerical data, fuzzy theory requires that the relationships be quantitative so that the modelling may be done. The MoI system, though applying fuzzy theory to the early design stages, does not explicitly address this.

Subsequently (Otto & Antonsson, 1993) the MoI method was advocated for application to *tuning parameters*, being manufacturing variation that can be adjusted (tuned) in response to design need. They motivated for the inclusion of tuning parameters in the design process using fuzzy sets, on the grounds that either the design can be made more tolerant of process variation, or the design can be allowed

to proceed with some uncertainty if it is known that the manufacturing process has sufficient tuning capacity to be able to accommodate the design.

Fuzzy theory is computationally efficient, much more so than the alternative of Monte Carlo, so that it can even be used as a paper based system under certain constraints. However fuzzy theory is also an approximate probabilistic computation method, and there is loss of information compared to Monte Carlo. In cases where uncertainty is very high anyway, this is perhaps less of an issue, and fuzzy theory does have the other advantage of being able to accommodate ranked qualitative parameters.

2.3.4 **Decision Analysis and Belief Networks**

Decision analysis is a means of systematically approaching decisions using probabilities. It provides a methodology to determine the likelihood of an outcome given uncertainty in the inputs. The main applications for *decision analysis* have been in business management, medicine, and environmental hazards. A comprehensive review of decision analysis is provided by Clemen (1996). His perspective is largely that of financial management and decision taking. Pellissier et al (1996) develop a model for assessing hip replacement decisions. They use the tree structure of decision analysis. Ridgman (1996) suggests that decision analysis and risk management can assist the process of new product development.

In decision analysis the problem is modelled as a *network* of decisions and chance events, see Figure 2.11. Each event is given a probability of occurrence: these are subjective probabilities for which the term *belief* is used, hence *belief network*. In most cases simple binary probabilities are given and not probability distributions. This is due to the simpler mathematics, and the difficulty in attempting a valid quantification of subjective belief. However in some cases the models have been constructed with probability distributions, for which the term *probabilistic reasoning* is used.

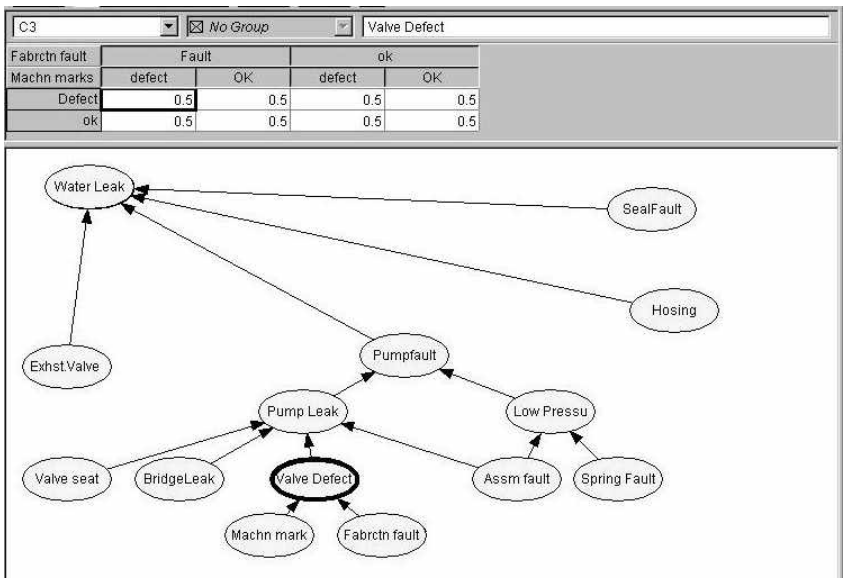


Figure 2.11: Bayesian belief network created with 'Hugin' to model pump faults.

Decision analysis is based on a *Bayesian belief network* which is a graph with nodes representing random variables that are dependent on each other. Each node may be in one or more states, and a probability of each of these states is provided by the user in the form of a table. The arrows in the graph show the direction in which the decision is made. *Influence diagrams* are belief networks with the additions of decisions and *utility* functions. A decision node shows a decision that is to be made by the user, and a utility node shows the costs (eg profit or loss). Figure 2.12 shows

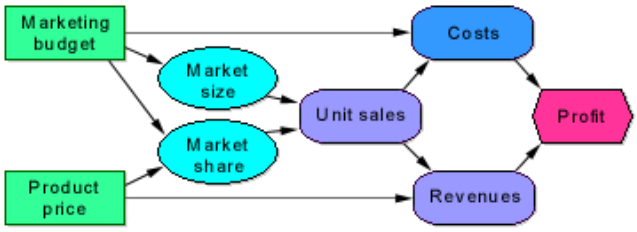


Figure 2.12: An Influence diagram shows decision nodes (i.e. variables that the user has control over, as rectangles), chance events that are uncontrolled (ellipses) and utilities (i.e. outcomes, as ovals). This diagram courtesy of Analytica (http://www.lumina.com).

an example. It is important to note that influence diagrams are not the same as flow charts or functional diagrams though they might look similar. An influence diagram shows the factors that are important in making a decision. Circular paths are not possible as they are in flow charts.

A *decision tree* is an influence diagram from another perspective. It displays the same information, but in a tree format. Decision trees show the alternative choices at points in the decision process. As one decision leads to another, the result is a tree structure. Decision trees show more explicit detail than influence diagrams, but can suffer from complexity and proliferation of nodes as a result. They tend to be used where there are discrete, usually binary choices at each decision, because they become clumsy when there are many branches to follow. Figure 2.13 illustrates a decision tree. Decision trees model *events* and their relationships, and not *components* and their functions.

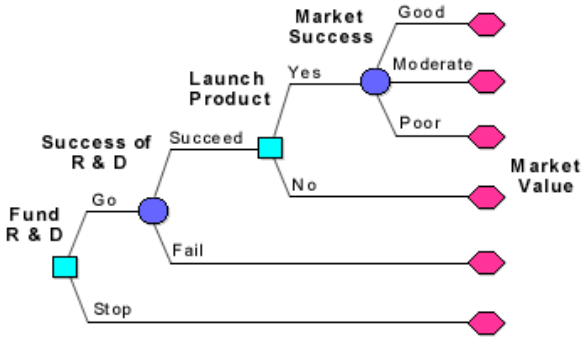


Figure 2.13: A decision tree for modelling market value of a product. The tree is read from left to right, and each split represents a decision or a chance external event. This diagram courtesy of Analytica (http://www.lumina.com).

The *Bayesian probability* method uses conditional probabilities, which are the probabilities that an outcome event has occurred when other events are known to have occurred. The difficulty with Bayesian probabilities is that it is often difficult to assign the required probabilities without resorting to arbitrary guesses or extensive testing.

Dynamic network models are an application of Bayesian methods with inference techniques. They are applied where variables fall into categories rather than being straight forward binary (yes/no, black/white). The systems quantify the variable dependencies and model them with probability distributions that may be normal or non-normal.

Decision analysis uses a combinatorial approach, as every state is combined with every other state for assessment. A significant disadvantage of this approach is the large number of assessments that have to be made when there are many states. These assessments are made manually, and therefore decision analysis usually involves only two or three choices at each stage. This limits their usefulness where many outcomes are modelled. Decision analysis has been extended to *qualitative possibility theory*, where the valuation sets are linearly ordered and that order is acted on (Dubois and Prade, 1999).

2.3.5 Constructing a Decision Analysis model

Existing decision analysis software

An example of commercially available software is *Hugin* (http://www.hugin.dk/). Hugin provides decision analysis and diagnosis. The user builds a graphical flowchart of the model with the mouse, and adds the relationships (beliefs or probabilities in the case of Hugin). The software applies Bayesian methods to determine the outcome based on the input probabilities assigned by the user. Hugin provides for multiple discrete probabilities. It also permits continuous probability distributions using the normal curve, but with certain limitations. Probability distributions other than normal are not supported. The main usefulness for a system like Hugin is in decision analysis, rather than in functional modelling. This is because decision trees model *events* and their relationships, and not *components* and their functions. Decision trees are essentially another form of *truth table*, a table of all the input variables and the outcome.

The starting point is to construct a diagram of the chance events and decisions. Typical applications are looking forward to predict likely outcomes of a financial investment, or looking backwards to determine the likely diagnosis for a particular fault condition.

The backwards diagnostic case is illustrated in Figure 2.11 and Figure 2.14, using the Hugin expert decision analysis system. The user first defines the events, in this case failure modes for a pump. The sealing failure could be due to either the Air Pump, Exhaust Valve, Hosing or Gasket. Each of these events can be developed to greater depth, and this is shown for the Air Pump (only). The user also defines a number of states for each event. For example the 'Valve seat' could be 'Warped seat' or 'Seat fine' (see the figure). Each of these states is then assigned a probability when the system is first set up, and a decision table. Multiple discrete probabilities are possible, as are continuous normal probability distributions. Figure 2.11 shows a decision table at the top, where the inputs and the output all take binary states, resulting in a decision table with eight cells.

Initially the belief network is set up with the nodes and their probabilities. To use the belief network on a specific problem, the user enters information that is known. There may be certain symptoms that are definitely present or absent, and these facts may be asserted in the model. This corresponds to forcing the selected probabilities to one or zero (or to intermediate values). Then the model may be run (propagated), and in doing so it produces the probabilities of the remaining events, as illustrated in Figure 2.14.

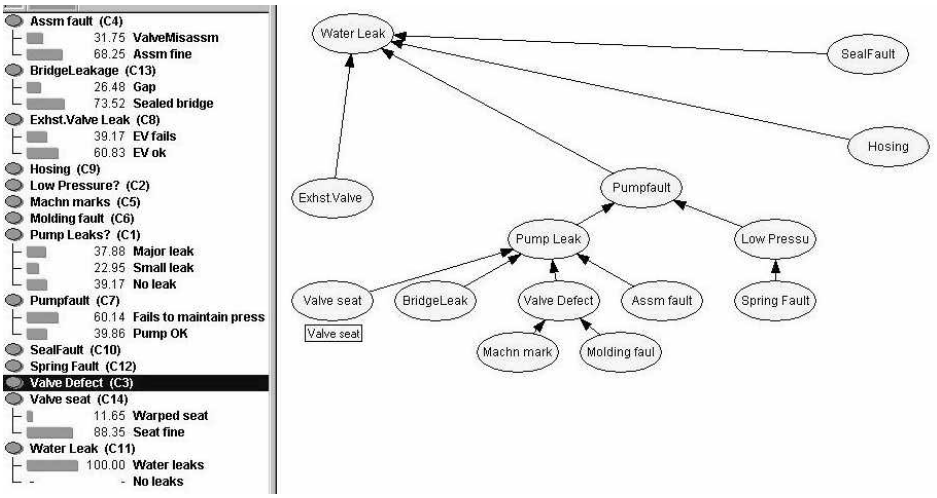


Figure 2.14: Decision analysis is applied to the failure of a pump, with the decision graph shown at right. In this example it has been asserted that the pump is leaking ('Water Leaks' = 1, at bottom left), and after propagation the likely causes of the fault are displayed in the list at left, along with their probabilities.

2.3.6 System engineering, management science, quantitative analysis, and operations research

In *management science* (also known as *operations research*, or *quantitative methods*), systematic methods are applied to solve management problems (Taylor, 1999). There are many methods within the discipline, though they all involve creating a model of the problem, as a set of mathematical relationships (including inequalities). A typical business problem might be to determine how many (volume) of various products should be produced to optimise profit, given fixed costs (plant costs that do not vary with volume) and variable costs (per unit). Once the problem is suitably formulated, then it is possible to determine the *break-even* point (the production volume at which total revenue equals total cost, i.e. zero profit). A commonly used measure of merit is *net worth* (total revenue less total production cost).

Linear programming

Many, if not most, problems in management science are set up with mathematically linear functions (y=mx+c) for the constraints, so that established *linear programming* (LP) methods may be used to find a *feasible region* (solution space). However, 'defining these relationships is typically the most difficult part of the formulation process' (Taylor, 1999, p 46). LP problems may be solved graphically when the constraints may be represented in two-dimensional space, but for higher dimensions the *simplex method* [46] is used. This is a numerical method that moves in the direction of greatest slope and is typically applied to parameters with continuous variables. It can also be used on some integer problems, though other methods exist that may be better for such cases.

[46] The simplex method allocates slack variables to represent unused resources, but then this results in more unknowns than equations, so simultaneous solution is impossible. The method avoids this by assigning zero to some variables, then finding a basic feasible solution, and then moving to adjacent solution points (Taylor, 1999, p164).

Characteristics of Linear Programming

Some characteristics of the LP method are:

- LP only checks the boundaries or extreme points in the solution space, because it may be shown that the optimal solution will always lie on the boundary. However the optimal solution thus found is also very close to violating a constraint. A small perturbation in operating conditions could push the system out of the solution space, and thus the optimal point is not necessarily *robust*. To compensate it is necessary for the analyst to apply sensitivity analysis, by adjusting parameters in the model and determining new optimal points.
- LP requires that the problem be deterministic, which suits most management science problems. However, when the parameters in the model are NOT known with certainty, then probabilistic methods have to be used, including decision analysis, Monte Carlo analysis, and Markov analysis.
- LP requires that the objective function be a single function and additive (either maximising or minimising a sum) (Taylor, 1999, p47).
- LP assumes continuous variables. In many cases the units of production are discrete. If the value of an individual unit is small then LP results may be rounded off without significant error. However if an optimal integer solution is required then it is necessary to use *integer programming* methods, eg the branch and bound method (Taylor, 1999, p 300).

Multi-criteria Decision Making

When there are multiple objectives, all of which need to be maximised, then the problem is termed *multi-criteria decision making* (MCDM). Two methods for MCDM are *goal programming* and the *analytical hierarchy process* (AHP) (Taylor, 1999, p 333). Goals are the various objectives, and goal programming is an extension of linear programming. To achieve this it is necessary to use weighting functions to indicate the relative importance of the goals. However using *weights* to combine dissimilar measures can be hard to defend when measures are on different scales, or

may even be qualitative. AHP also develops a numerical score for each criterion, based on pair-wise comparisons that are rated on a preference scale (eg 1 to 9).

Non-Linear programming

When the relationships are not linear in the variables, then finding a solution is more difficult as the solution is not necessarily at a boundary, and it is necessary to use *non-linear programming*. Hill-climbing algorithms such as the simplex method will not necessarily find the point of global optimisation. When non-linear problems contain only equality constraints then substitution and differentiation may be used (Taylor, 1999, p 391). A more flexible method is that of Lagrange multipliers though 'the mathematics become overwhelmingly difficult' for large problem (Taylor, 1999, p 395).

Transportation problems

The management science methods are also applied to *transportation* problems. The characteristics of this type of problem are that there are several sources and destinations (each with fixed supply and demand). The goal is to find the minimum cost of transporting the product (Taylor, 1999, p 233). There are solution methods other than the simplex method for transportation problems. *Assignment* problems are similar to transportation problems, but the supply and demand are only one unit each, an example being personnel assigned to tasks.

Network analysis

A *network* is set of multiple interconnected paths (branches or arcs) along which action occurs. Examples include road systems, production processes, project management schedules, and computer networks. The paths are connected at nodes (vertices).[47] Each branch has a property assigned to it, such as time or cost. There are three common problem types: shortest route (minimise the time or cost to traverse the network), minimal spanning tree (connect all nodes but minimise total

[47] An example of a networks is a system of roads. The roads are the branches and the nodes are cities. Each road (branch) has a value of distance (or travel time) associated with it.

branch length), and maximal flow (maximise the total flow through the network given branches with limited flow capacity).

Simple networks may be solved using linear programming methods, providing that the arc properties are constant, deterministic, and steady-state. More complex networks are addressed using PERT and the related critical path method (CPM).[48] The critical path is the longest time path through the network. These methods originated with project management but have since been applied to other domains. However PERT provides only an approximate probabilistic computation method as it includes significant simplifications: it relies on moments of the distributions, assumes that the result is a normal distribution, and is limited to mathematical operators of addition and subtraction.

Graphical evaluation and review technique (GERT)[49] is an extension of PERT. It is a procedure for analysing networks where the arcs are described by two (or more) parameters. The first parameter corresponds to the single parameter of conventional network analysis (eg time taken to traverse the arc) and the second is the conditional probability that the arc is traversed (Pritsker, 1990). GERT derives the probability that a node is reached, and the moment generating function of the time between nodes. The analysis is tractable for model that only contain 'or' nodes. [50] The method requires that variables associated with arcs be added, and functions such as multiplication and minimum are not supported. For full probabilistic computation of network performance it is necessary to use a method such as Monte Carlo analysis instead.

---

[48] CPM and PERT are effectively now a single method, but originally CPM used deterministic values with the activities as the nodes, while PERT used probabilistic values and the activities as arcs (Taylor, 1999, p 452).

[49] There is nothing notably graphical about the method. This is because GERT was first named for Pritsker's mother, and afterwards the G was assigned to graphical (Pritsker, 1990, p 240).

[50] 'No computationally feasible method of analysis for *and* or *Inclusive-or* nodes has been developed for times which are random variables' (Pritsker, 1990, p217).

Markov analysis is somewhat related to network analysis, but it is used where systems can move from one state to another (eg a machine may become broken) with a known (and constant) transition probability. Simple state transitions may be analysed using decision trees, but Markov analysis provides matrix algebra tools that are more effective. The analysis returns the steady-state probabilities of the system (Taylor, 1999, p 602-618).

Queueing problems

Dynamic simulation of networks is a demanding aspect of systems engineering. Typical problems for industrial engineers and management scientists are queues (eg multiple bank tellers serving a queue of customers), and there is a significant body of knowledge on *queueing theory*. The required output is typically the average wait time for service, providing that average service rate is faster than customer arrival rate so that an infinitely long queue does not develop. However the difficult aspect is the randomness of the events (the delay at servers and the arrival of customers). Explicit solutions are available for certain cases, typically arrival rate with a Poisson distribution, and service times that are Exponentially distributed. For other suitable networks it is possible to use Markov analysis or an extension of GERT (Pritsker, 1990, p246). However for more complex networks the explicit solution approaches are infeasible and Monte Carlo analysis is necessary (Taylor, 1999, p 408).

System engineering

System engineering (or system analysis) incorporates a number of methodologies. Some see it as the "structured and disciplined practice of top-down function-driven engineering development that provides a traceable and verifiable record of all engineering decisions" (Mar, 1991). Others use it as for evaluating solutions and *risk*[51] (Ossenbruggen, 1994). It also overlaps with management science as it concerns the *optimisation* of a solution given multiple simultaneous constraints.

Like the design process, systems analysis involves problem definition, generation of alternatives, model formulation, and analysis & alternative selection (Ossenbruggen,

[51]Ossenbruggen (1994) defines risk as the product of monetary consequence and probability.

1994). Optimisation models use the quantitative mathematical functions of management science. In particular the model formulation stage involves a *decoupling* process analogous to the decomposition process in design. The term *feedback analysis* is used to refer to the exploration and refinement of the model.

Probability distributions (in the form of Normal mean and standard deviation) may be incorporated into systems analysis. The mean and variance of the net worth may be determined from those of the inputs[52].

Both Systems Engineering and Management Science are concerned with optimisation, and they use the same tools, so they are not further differentiated here. Therefore the term 'management science' as used above should be understood to apply to system engineering too.

Management science is also called quantitative analysis, which confirms that the methodology does not accommodate qualitative models easily.

2.3.7 **Monte Carlo simulation**

Monte Carlo is a quantitative simulation method. It applies mathematical modelling, and permits the inputs to be represented by probability distributions. It randomly samples from these inputs, applies the mathematics to compute the result, and then repeats for different random samples. Once this process has been repeated sufficient times, a probability distribution may be created for the output parameter.

The standard Monte Carlo method is to generate a random number between zero and one. This corresponds to the cumulative probability that is of course also zero to one. Then the explicit inverse probability function is used to determine the time value.

---

[52] Given input X_1 with mean μ_1 and standard deviation σ_1, and similarly a second input X_2 with μ_2 and σ_2, then a net worth given as the weighted sum $Y = a_1.X_1 + a_2.X_2$ (where a_1 and a_2 are constants) will have mean $\mu_Y = a_1.\mu_1 + a_2.\mu_2$ and variance $\sigma_Y^2 = a_1^2.\sigma_1^2 + a_2^2.\sigma_2^2 + 2.a_1.a_2.\sigma_1.\sigma_2.\rho_{12}$ where ρ_{12} is the correlation coefficient (which is zero if X_1 and X_2 are independent).

A disadvantage of this approach is that the input random number has equal probability of occurring over the range zero to one. This means that the result will be time values that occur as often in the tails as in the centre of the distribution. This is contrary to the requirement for a distribution with *fewer* values in the tails. Consequently the standard Monte Carlo method does not reproduce the shape of the probability distribution very well (Vose, 1996).[53]

An improved Monte Carlo method is *Latin Hypercube Sampling* (LHS). This splits ("stratifies") the probability distribution into bands of equal cumulative probability. The time bands are therefore wider towards the tails. One of these probability bands is randomly selected. Then a second random number selects a probability value inside that band, and the time value is calculated from that. The band is then marked as used so that it is unavailable for further selection. This ensures that all bands receive coverage eventually.

The Monte Carlo method generally uses probability distributions that may be expressed as an algebraic relationship for the cumulative probability, and which may be rearranged to give the inverse function, namely time as a function of cumulative probability. However algorithms exist whereby the method can accommodate histogram inputs (Manno, 1999). An example set of Monte Carlo screens from a commercial modelling system[54] is shown in Figure 2.15.

---

[53]Vose (1996) describes the existing body of knowledge regarding quantitative risk analysis using Monte Carlo simulation. The book describes (among other topics) Monte Carlo and Latin Hypercube sampling, a comprehensive list of probability distributions, fitting a distribution to data and expert opinion, and methods to accommodate dependencies.

[54]*Existing Monte Carlo analysis software:* The Palisade company (http://www.palisade.com) have a commercial risk analysis product called "@RISK". This provides risk analysis functions within a spreadsheet (Microsoft Excel®). The user creates a conventional spreadsheet containing mathematical expressions. Next the user identifies which values in the spreadsheet are subject to variability, and replaces them with an @RISK probability function. Numerous distributions are supported including PERT, Exponential, Poisson, Binomial, Extreme Value, Lognormal, Chi-Square, Gamma, Student's t, Normal, Uniform, Discrete, Weibull and others. The software uses then uses Monte Carlo simulation to compute the outcome probability distribution. A similar Monte Carlo simulation system is provided by Crystal Ball. Yet another system is *Analytica* (http://www.lumina.com/). *Analytica* differs from the others in its user interface. In particular it has a graphical interface on which the user creates an *influence diagram*. However, unlike other influence diagram users (such as Hugin), it does not use decision tables but quantitative mathematical formulae. The user assigns a probability distribution to each of the inputs and then the software uses Monte Carlo analysis to determine the outcome probability. Typical uses are business modelling.

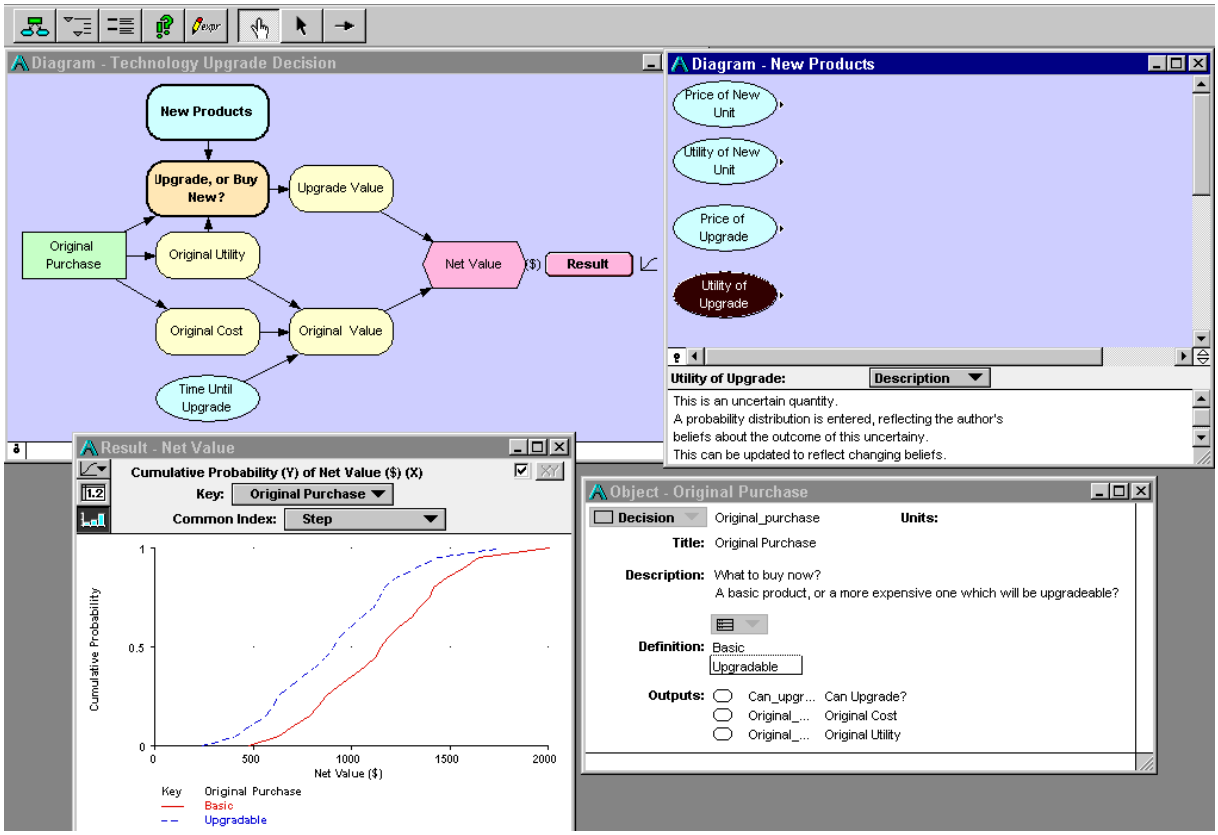


Figure 2.15: Analytica is a decision analysis tool that is used to model risk events. This tutorial example models a purchasing decision: whether to buy a product that can be upgraded, as opposed to the basic product. The model is shown at top left, as a graph connecting various decision and chance events. The user provides the relationships that describe the graph, and the probabilities of various outcomes, and the system calculates the final result. The result in this case is present value, and this is shown by the chart at bottom right (cumulative probability). The details of one decision event are shown at top right and bottom right.

Siu (1990) notes that with the *Bayesian* methods used in probabilistic reasoning, it is difficult to find a formal solution because of the large number of variables and the uncertainty in the data. The uncertainty has to be described explicitly by a probability distribution, and this is a burdensome process. Siu therefore used Monte Carlo methods for estimating the uncertainty.

2.3.8 Qualitative simulation

Qualitative simulation predicts qualitative behaviour from a given qualitative differential equation and initial state. The *differential equations* are solved (by constraint satisfaction or related methods) to give solutions that are bounding *intervals*. The methods may be used for qualitative dynamic or equilibrium performance. Semi-quantitative reasoning (eg ‘Q2' and’ Q3', Kuipers 1994) is an extension of qualitative simulation to include incomplete *quantitative* knowledge. The incomplete quantitative variable is assumed to be a member of a bounding interval (i.e. a set given simply by a lower and upper limit) or alternatively a *fuzzy set* or probability distribution. Applications include Wang and Chen (1995).

Qualitative simulation can be viewed as an extension of ordinary differential equations towards more generality. Though the ability to span the qualitative - quantitative divide is useful, and they can be used to simulate dynamic systems, the limitation of the method is that it requires that the problem be formulated as a set of differential equations. In the case of supporting the design process, this could be a major limitation as design problems are seldom expressed, and may not even be expressible, in terms of differential equations.

Qualitative simulation is perhaps an overstatement, since the methodology is only able to operate on ordered qualitative scales, and not on nominal one.[55]

[55]The interested reader is referred to Chapter 6 for a description of how the terms quantitative and qualitative are used in this work.

2.4 Assessment constraints

Having an assessment mechanism may be a necessary part of assessing a design solution. However there also need to be constraints or criteria to measure against. This section reviews constraints that apply to the design process.

2.4.1 Quantitative assessment criteria

Any qualitative modelling or simulation system requires appropriate criteria against which to assess the outputs. From the functional modelling perspective Pahl and Beitz (1988) put forward three primary general objectives of functional modelling, which may be re-interpreted as assessment criteria: *technical feasibility*, *economic feasibility*, and *safety* requirements. Furthermore they list additional constraints that a solution must satisfy, namely reliability, ergonomics, production methods, quality control, assembly, transport facilities, intended operation, and maintenance. They advise that these constraints be considered during the concept stage, “at least in essence”. Interactions that are listed for checking in Raine (1998) are:

- Functional performance and integrity
- Compliance with design specification and response to customer need
- Geometric and kinematic constraints
- Material selection
- Dynamics, stress and deflection
- Form features for manufacturing
- Tolerance stacks
- CNC machine code
- Costing
- failure mode effects and criticality analysis (FMECA)
- Manufacturing implications

In principle these constraints are valid for any modelling or simulation system.

Performance is difficult to measure when the design is still at an abstract stage, unless the designer instantiates a solution at a less abstract level. Schmidt and Cagan (1993) see machine performance metrics as including size, weight, power, efficiency, capacity for force generation, and economic features, but that precise metrics such as these are “difficult to articulate for designs that are defined as function structures”. Functional modelling is often a visualisation method rather than a simulation system, and therefore does not produce outputs of itself. While it is helpful to have checklists, such as the above, they are subjective and difficult to reconcile with the output of any modelling or simulation system. Functional modelling itself is silent as to how constraints (eg ‘ergonomics’) are to be applied. Pahl & Beitz (1988) list some methods that can be used to support the process, namely:

- *The method of persistent questions*: Checklists and questionnaires are used as a stimulus to thought and creativity.
- *The method of negation*: This is a method of “*systematic doubting*”, whereby individual components or functions in the system are negated, to explore alternative possibilities.

However the underlying difficulty of reconciling constraints against simulation output remains as a limitation of functional modelling and indeed of many simulation systems. The problem is that the constraints are frequently qualitative, whereas the simulation results are quantitative. In the case of modelling for problem visualisation there are no simulation results at all.

2.4.2 **Measurement scales**

One way of solving the problem of assessing quantitative results is to use a measurement scale. It is relatively straightforward to define measurement scales when the objectives of a decision are purely in one easily-measured unit such as dollars. However it becomes more difficult to find a solution where multiple objectives exist on different scales, or where one objective (goal) is subjective. This is part of the *multi-criteria decision* problem in engineering design, economics, management science and other disciplines. Several methods have been proposed and used,

including QFD, AHP, Pugh, multi-criteria optimisation and decision theory (Scott and Antonsson, 1999). All such methods aggregate results to produce a single metric for decision making. The methods available for defining measurement scales are as follow.

Trade-off weights

This method uses proportional scoring to rearrange the scale of the objectives, so that both are out of 100 (for example). Then *weights* are assigned according to the relative importance of the objectives. An objective that is more important will be given a higher weight. Usually the weights range between zero (no importance) to one, with the sum of all the weights being one. The proportional score is then multiplied by the weight and the total score determined for each alternative. Weighting the objectives permits different objectives to be combined, but the method typically relies on the subjective assessment of the weights and therefore on opinion.

Present value

In any decision analysis it is necessary to decide how far ahead to look (the planning horizon), and how to value the consequences. One useful method of valuation when the consequences are entirely financial, is to determine the *present value*, which is the future value referenced back to today by the *discount rate*[56]. Present value is given by

$$Pv = \frac{x}{(1+r)^n}$$

where

x monetary value at year n

r_i interest- or discount-rate

n number of years to when value x will be received

[56]The discount rate is the acceptable rate of return, and therefore varies with circumstances. It is usually the bank interest rate plus a loading that reflects the risk and uncertainty. *Internal rate of return* (IRR) is the breakeven rate at which net present worth is zero.

A typical business decision may involve some expenses spread over some years, with returns also spread out and perhaps overlapping with the expenses. In such cases the *Net Present Value* (NPV, also called net present worth) is determined:

$$Npv = \sum_{i=0}^{n} \frac{x_i}{(1+r)^i}$$

where

x_i income at year i (negative values may be used for expenses)

and other parameters as defined above.

A commonly used measure of merit in *systems analysis* is *net worth* (total revenue less total production cost)[57]. For public sector investments the benefit may be evaluated as savings for the public good [58].

A *weak order*[59] is one that depends on a simple comparison and produces an ordinal scale with no numerical values (Scott and Antonsson, 1999). Weak orders are qualitative and can be difficult to assess. Scott and Antonsson state, 'any computational method for decision-making requires the further structure of a numerical scale that ranks alternatives' (p219) which is called a value function, and is always possible to construct from a weak order. They identify the difficulty that:

---

[57]Ossenbruggen (1994) states that: "A primary purpose of a capital investment is to improve productivity" (p 120), measured as net worth. Capital investment decreases production costs, so that either a greater profit (net worth) may be made by selling product at the existing price, or the sale price may be lowered and market share increased (in which case greater volumes are expected to lead to increased net worth).

[58]Many systems analyses use net present worth as the criterion to be optimised. For public sector investments the benefit may be evaluated as savings for the public good. *Supply* and *demand* functions are defined, relating price to quantity, usually with exponential functions. Elastic (inelastic) demand is said to occur when the price is affected in a minor (major) way by changes in quantity. Then the *direct user benefit* may be calculated as the area under the price-quality curve due to a change in the supply curve. The discount rate for public projects could be lower than the bank interest rate, as low as or even lower than that of government bonds.

[59]The interested reader is referred to Chapter 6 for a description of how the terms quantitative and qualitative are used in this work.

> *'the correct specification of the numerical scale is crucial to the satisfactory resolution of both the multi-criteria decision problem and the problem of decision under uncertainty. .. If the comparison of preferences is effected by the arbitrary assignment of numbers to alternatives, then those preferences contain no more information than weak orders. ... In that case no aggregation method is adequate. .. A decision method must have an explicit procedure for assigning values to alternatives and for combining those values into a single performance function, and the two must agree*' (p219, 226, 227)

There are many systems used in design (eg weighting methods, QFD, fuzzy theory, optimisation, genetic algorithms) that are vulnerable to the effects of weak orders being used as the foundation for decision making. The problem of weak orders is a significant one in all systems that attempt to automate the decision making process, because parameters may be subjective or difficult to establish. Engineering design is a decision making process too, and the weak order problem is significant in the poor performance of automated design systems. Given the subjective issues, many of them not even resolved into decision criteria, there are major challenges in developing adequate assessment criteria for anything but the most simple design decision problems.

2.4.3 **Assessing qualitative outputs with bench marking**

Bench marking is a comparison of the company's product or process against those of others. One form is bench marking against competitors. Their products and service are analysed and used as the basis of comparison. Bench marking can also be a comparison of a process at the company against that of another company that is not a competitor. For example the way that incoming sales enquiries are handled could be compared. These types of processes are used universally, so any other company

that performs the process well can be used for bench marking. Using a company that is not a competitor also results in a greater willingness to exchange ideas. Bench marking may also be done internally, by comparing one department against another.

The objective of bench marking is to find the best practice. This is used for improving the quality of the product and the processes that produce it. It is not so much that the benchmark is used for copying, but used for internal reassessment. Some features of the process are illustrated in Figure 2.16.

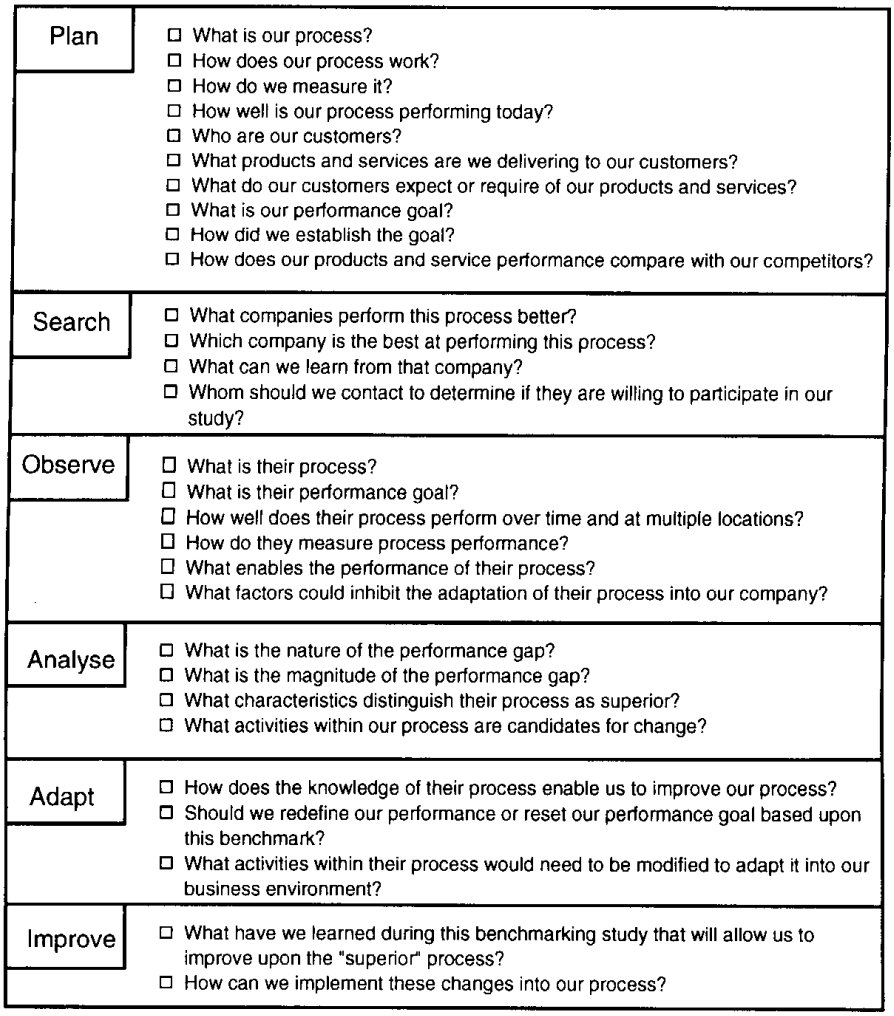


Figure 2.16: Checklist for bench marking. Attributed to Watson in Bergman & Klefsjö (1994).

2.4.4 Other qualitative assessment strategies

Zhong and Dooner (1996) present a prototype system that qualitatively evaluates mechanical fasteners. The parameters under design control are the type of feature (thread, head type, etc.), material, installation, and performance. The system consists of a library of pre-defined features and associated rules. In use the designer selects features or functions, and the system shows the approximate appearance of the configuration. The result is sent for a qualitative assessment. The assessment modeller uses qualitative rules (heuristics) to predict performance such as ‘pull out resistance’ in terms of adjectives (good...bad). The system is domain specific and the necessity for closely defined qualitative rules would tend to suggest that it might be difficult to scale it up to model function more generally.

2.4.5 Assessing multiple viewpoints

Some design models (eg Raine et al, 2001) specifically address the need to check the solution from multiple performance viewpoints, including function, geometry, kinematics, material, etc. The Raine model is illustrated in Figure 2.17 below.

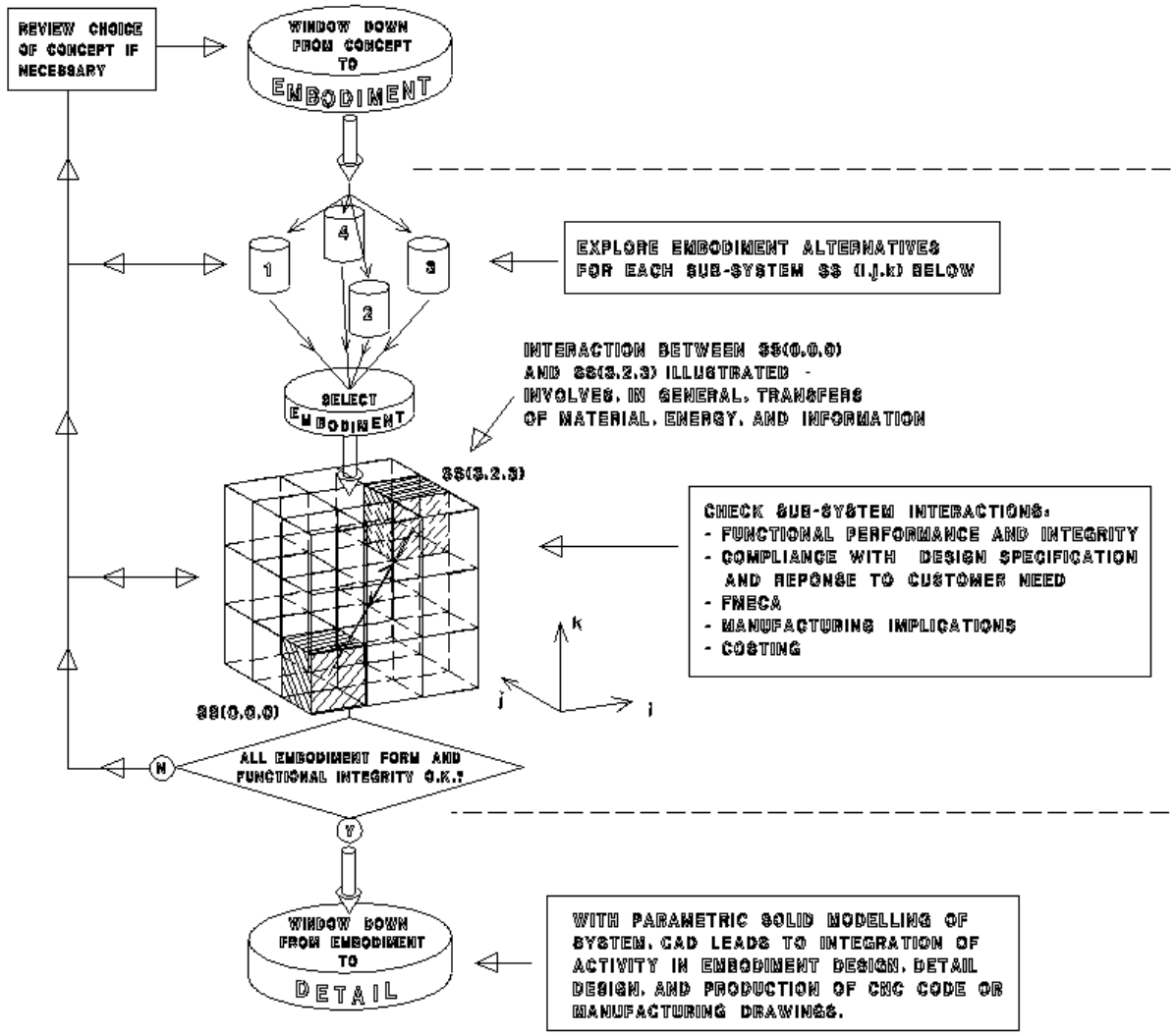


Figure 2.17: Raine's model of recursive exploration of alternatives. This model specifically addresses the issue of checking the design from multiple viewpoints, and is a development of Pugh's Total Design activity model (Pugh, 1991) with concurrent design, manufacturing planning and market development, plus increased focus on customer needs and the market. Adapted from Raine et al (2001) with some detail omitted for clarity.

The model anticipates a simulation based on energy, material and information flows between sub-systems in the machine. However it is a model of the design process and of itself does not provide mechanisms to achieve that multi-viewpoint checking.

2.5 Decision mechanisms

Every design process involves making decisions at some point. One solution has to be selected over other candidates. Generally the decision is made by humans, though there are mechanisms that can assist the process, and these are reviewed here.

2.5.1 Belief functions

In an interesting application of *belief* functions to engineering design, D'Ambrosio and Ullman (1995), and Herling et al (1995) studied the decision making process as applied to engineering design. They observed that there were few decision support tools available to assist designers, especially during the early conceptual stages, and furthermore that existing methods were of limited effectiveness. They identified some causes as the inability of existing decision theory methods to accommodate the type of information that is typical of early design phases: qualitative, incomplete, changing, and inconsistent.

Their strategy involved developing computer tools, based on decision theory taxonomy. They modelled the design process as an influence diagram, in terms of:

- *Issues*: the need to meet a design specification, shown as a decision node
- *Alternatives* (proposals): different solutions to meet the *issue*, shown as input values for the *decision node*
- *Arguments*: rationale that is given to support or oppose an *alternative*, shown as expected *utility*. *Arguments* are modelled in terms of:
    - *Criteria* against which an *alternative* is to be judged, based on acceptable or unacceptable. Criteria are requirements, specifications or constraints.
    - *Knowledge*, level of expertise of individual designer based on a descriptive scale: (expert - experienced - informed - amateur - weak - unknowledgeable)

- *Confidence*, the extent to which a belief is held, in particular the belief that the proposed *alternative* will satisfy the *criteria*. They use a descriptive scale: (perfect - likely - potential - questionable - unlikely).

They then determined the value for each *alternative*, as the product of the *criteria* for an *alternative* and the probability of its occurrence. The probability was determined from the individual's *knowledge* and *confidence* factors, for each participant in the design team, and the sum determined the expected value of the *alternative* in question. On this basis each of the *alternatives* could be assessed for expected value.

Although they did not explicitly make the link, this concept of knowledge as separate to confidence is similar to *Dempster-Shafer belief functions* which partition uncertainty into two parts, which are evaluated separately. The parts are
(i) *support*: the extent to which the evidence supports the assertion, and
(ii) *plausibility*: the extent to which the assertion is plausible (Biondo, 1990).
The method can deal with partial beliefs, as well as ignorance. For example, an assertion might be that a pump can fail due to dirt on the valves. There might be only limited evidence to support this (eg 0.6), but the assertion is highly plausible (0.9). The belief function for the assertion would be (0.6,0.9).

A limitation of the method of D'Ambrosio et al concerns the underlying utility theory. Advocates of utility theory characteristically maintain that all criteria can be expressed quantitatively, and can be combined using weights to obtain one final combined measure of utility, commonly in monetary value. However this is not always realistic, as the weak order of many qualitative variables makes it difficult to defend the quantification process and the weights assigned. Consequently the decisions made on the basis of the final measure of utility can be compromised.

A benefit of their approach is that the effect of *design teams* may be modelled. Each member of the design team provides weighting to the decision in proportion to their *knowledge* and *confidence* regarding each *alternative*. The set of criteria on which a

design is judged is not generally the same for all members of a team, and the method accommodates the conflicting assessments that may arise. The method also accommodates uncertainty and incomplete information. It also provides a means of recording the *rationale* behind a design decision (viz. *design intent*). They acknowledge that some extreme simplifications have been implemented in the underlying decision theory, but motivate this on the grounds of needing to minimise the burden of representing each decision node. They plan further work to validate the system, and improvements to address graded criteria (*quantitative*) scales, and the difficulties with interpreting descriptive scales.

2.5.2 Classification of decision problems

Ullman and D'Ambrosio (1995) describe a taxonomy of engineering decision problems. They describe various attributes that a design problem may have, from the perspective of decision making. The classification system is stated to be intended for both design problems and decision support tools. The main features are shown in Figure 2.18 and Table 2.1.

1. Problem completeness	Refers to how well the issues are known • Complete: all issues are known and the criteria are fixed • Incomplete: issues and criteria are unresolved or weakly understood, not all alternatives have been developed
2. Abstraction	The type of information that is available • Quantitative: refined information is available • Qualitative: abstract information is available, alternatives and criteria may be unquantifiable • Mixed: both kinds of information are available
3. Determinism	Refers to the range of information • Deterministic: parameters take on point values • Distributed: parameters vary according to some probability distribution, which itself may be unknown
4. Objective function	This is the method of evaluating how well the alternative meets the criteria, or in decision theory terms, the utility or cost. • Optimum: a maximum, minimum, or other optimal point is sought • Judgement: a weighted preference across many parameters is used
5. Consistency	Refers to the unity or conflict between the preferences that various team members hold. • Consistent: there is only one set of assessment criteria. This exists where there is only viewpoint, or all viewpoints are unified in outlook. • Inconsistent: there exist other viewpoints, which conflict with each other.
6. Comparison basis	• Absolute: alternatives are compared to an absolute criterion • Relative: alternatives are compared to another alternative

7. Dimension (of certainty)	This refers to the means to express certainty, and assumes that there are two dimensions to certainty, one being a measure of the confidence of an alternative meeting the criteria, and the other being the depth of knowledge about the alternative. Both confidence and knowledge are assumed measured by a probability. • None: there is no measure of either confidence or knowledge • One: either confidence or knowledge is measured • Two: both confidence and knowledge are measured
8. Belief completeness	This refers to the team involvement in the decision. • Complete: all teams members are able to give an evaluation on every alternative. • Incomplete: not all alternatives are evaluated by every team member.
9. Problem focus	Design issues may refer to one of the following • Product: technical issues regarding the product itself • Process: planning issues regarding design, manufacture or distribution
10. Range of issue independence	Refers to how design issues are related to other issues • Type I Independent issues: the design issues are totally independent of each other, and may be solved separately. This requires that the design problem does not change with time, that one design problem does not affect another, nor must one decision ever wait for another. Decisions are not retracted. • Type II Dependent issues: this is the dependence of one design issue on another single issue. Alternatives are shared by more than one design issue. Some design issues have to be resolved before other ones can proceed. Decisions that are retracted affect other design issues. • Type III Inter-dependent issues: design issues are dependent on multiple other issues , including issues above and below it.
11. Level of support	This refers to the type of support that is provided towards decision making. • Representation: this is the lowest level of support, and is simply the statement of issues, alternatives, arguments and criteria. • Outcome determination: a level of support which allows the outcome to be determined. This may be achieved by tests or simulation. • Decision analysis: at this level of support the system is capable of selecting an alternative.

Table 2.1: Classification of decision problems, adapted from Ullman and D'Ambrosio (1995).

The classification scheme shows the complexity of engineering design decisions. They apply the taxonomy to several decision support methods, including decision trees, Pugh's method, and utility theory among others. They observe that engineering design teams frequently make decisions poorly, take too long, rehash old decisions, drop issues, and not record reasons for decisions. They conclude that available decision support tools are limited in their ability to support the engineering design process in all its complexities, but that even the simplest assistance in structuring the design problem can be valuable.

Comments on decision making in design

The need to establish assessment criteria has special relevance to simulation systems. In developing or assessing these systems it might be valuable to consider whether the 'Objective function' is to optimise a single parameter (or several independent parameters), or whether judgement is needed to decide between many parameters. Ullman and D'Ambrosio list only these two objective functions. They interpret judgement as a weighted preference, and by implication assume that all

criteria and simulation results are necessarily quantitative. Invoking the earlier weak order discussion, it may be appropriate to add another type of objective function, namely qualitative judgement.

Engineering design decisions are apparently often in the more complex categories of the taxonomy, with incompletely known criteria, quantitative and qualitative information, parameters that are random variables, unclear objective functions & relative comparison of alternatives, inconsistencies & uncertainties within the design team, and dependencies between design issues.

The latter is the distributed design problem, where functions cannot be decomposed into independent units. The design process is sufficiently well understood to realise that the distributed design problem does occur, and both human designers and automated design tools find it difficult to handle.

There are many assessment decisions that the design team has to make in the areas of 'Consistency', 'Belief completeness', and 'Dimension of Certainty'. The reality is that criteria are likely to be inconsistent, with team members differing in their technical value systems. Furthermore the team evaluation is potentially incomplete, with some team members making no contribution, perhaps

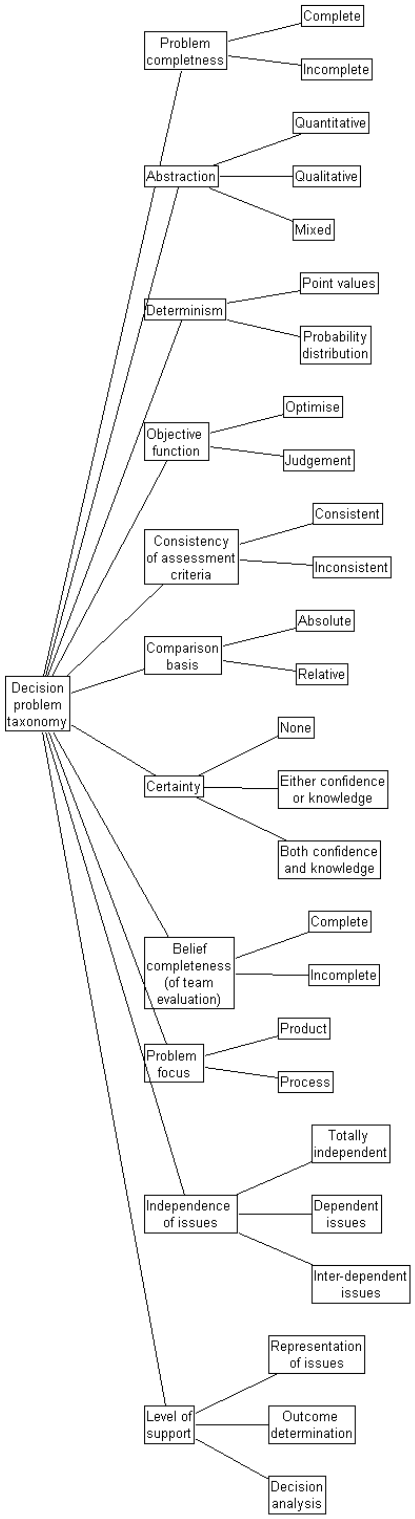


Figure 2.18: Taxonomy of design, adapted from Ullman and D'Ambrosio (1995), showing possible attributes of a design problem from the perspective of decision making.

due to personality dynamics among other potential causes. Team values are also determined by the certainty of assessment, which in turn is determined by the knowledge that individuals have of the subject area, and the confidence of their opinion. Issues that can be anticipated are those of personality, authority structures, and technical dominance of some individuals over others.

It is by no means apparent that designers consider this multitude of factors, even informally. Design support tools, including artificial intelligence, are probably even further away from being able to accommodate these effects. Support for this view is found in Bartsch-Sporl and Bakhtari (1996) who believe that artificial intelligence methods are incapable of providing a fully automated assessment of the multiple aspects that determine the quality of the product, and that assessment criteria are not universal, but are partly determined by higher strategies.

The ‘Level of (decision) support’ is another topic of interest. This concerns the expectations of the assessment: whether it should merely represent the issues (i.e. present the facts and leave all the decisions to someone else), or determine the outcome, or at the highest level make a decision. These decisions apply both to human decision mechanisms (the designer may be required to simply represent issues to management or make a decision himself) and to artificial intelligence design tools. Although many artificial intelligence projects attempt to provide complete decision making, such a strategy is of doubtful value in winning human support from designers themselves. Furthermore, the status of fielded artificial intelligence applications suggests it is by no means proven that artificial intelligence systems can deliver valuable results at this level of support other than at trivial design tasks. More modest objectives from artificial intelligence systems have latterly been proposed, eg Bracewell et al (1996a), with possibly more successful outcomes.

2.5.3 Conflict resolution

Conflicts may arise during the design process, and these affect the assessment process. An assessment remains controversial while conflict exists. Conflicts may arise from the design process itself, in particular *concurrent engineering* has the potential to reduce developmental times, but complicates the management of the process. Another source of conflict is *distributed design*, and the consequent inability to decompose the design problem and solve it piecemeal causes conflict with the design team and potentially to other processes such as manufacturing. The problem in both concurrent engineering and distributed design is often solved by a management decision: one part of the problem is given priority, and the rest of the design has to accommodate to that (Medland, 1996).

Conflicts also arise from:

- differences in knowledge of group members
- differences in confidence (experience) of group members
- design criteria are contradictory
- design goals differ between team members
- technical bias and prejudice of the designer
- different basic assumptions
- different technical vocabulary between specialists
- different assessment criteria
- conflicts of human personalities that occur due to hostility, although these are outside the immediate technical domain
- operating principles may conflict with design constraints or objectives

While conflicts may be inevitable, they are not necessarily destructive. However they need to be managed so that they do not destroy the integrity of the design process.

The topic of semi-automatic conflict resolution in design has been addressed in several research developments, some of which are reviewed by Oh and Sharpe (1996). Common themes in conflict resolution strategies are the use of:

- independent *agents* each of which attempts to optimise its own agenda (viz. *blackboard*)
- third party intervention from an arbitrator
- *multi-criteria decision making* (MCDM)
- weighted sums or indices
- heuristic rules
- negotiation in the form of constraint relaxation

Oh and Sharpe (1996) see one of the benefits of a structured approach and the partly-automated generation of solutions as a limitation to the designer's belief and bias being imposed on the design. However a counter argument might be that such an approach risks limiting the discovery of a truly novel solution, especially since solution generation systems tend to be very much less than omniscient.

Managing design in multiple user environments

Medland (1996) explores the decision making process in engineering design, with a *database* system that controls who has rights to change design elements. The method involves decomposing a problem into sub problems, and encoding the design in terms of a set of constraints or rules. Unresolved conflicts between the proposed design and the rules are reported up the hierarchy. The system uses token handling, in which tokens correspond to rules that are satisfied. Tokens are fed up to higher levels, and the system thereby attempts to find a state in which all rules are satisfied. A prototype system is built around a test case of configuring the design of a ball point pen. The rules used are constraints on dimensions of the geometry, various lengths and diameters.

The test case is relatively simple, perhaps even trivial, but the ideas are interesting. The actual mechanism whereby the system tries to find a solution state is not explained, and it appears that the system is passive in this regard. Apparently the system does not generate solutions, this being left to the designer. Instead it is a system for evaluating designs against a set of pre-existing rules. Its limitations can be expected to be in the formality of approach, namely the reliance on functional decomposition (being a common limitation of most design support tools), and the

necessity for rules to be identified and coded into the system. The system for operating the rules is not described, other than it being a database, though it bears greater similarity in function to an *expert system*. The stated intent was to manage distributed design constraints, but apart from identifying the needs, it is not apparent from the presented material that this has been achieved.

One of the needs identified by Medland is for a control (management) system that permits different users to modify the sub-parts of the design under their control. Another aspect of management is that of managing conflicts that have come up from different sub-problems. The expressed need was for a higher level to make decisions whether to accept or reject the proposed design, and thereby force one of the sub-problems to find an alternative solution. Neither aspect of management appears to have been achieved in the prototype system presented by Medland, other than perhaps in a passive way, though perhaps subsequent work may address these interesting aspects of the design process.

Engineering design decisions have been compared to the *social choice* problem of voting by Scott and Antonsson (1999). They note that expertise is distributed throughout a design organisation, and group decisions are made. They discuss whether making a design decision is a multi-objective decision or a social choice problem and argue for the former since ‘in engineering .. attributes, not people, must be reconciled’ (p226).

2.5.4 **Risk analysis**

Risk analysis is a prominent form of decision mechanism, though it is more a group of different mechanisms than a single approach (Heilmann, 1990). As will be shown, risk analysis addresses qualitative and quantitative aspects.

Risk refers to possible loss, including undesirable outcomes such as failure modes, loss of life, injury, financial loss, project duration, machine reliability. The term ‘risk’ is

used somewhat ambiguously: some use it as a probability, while others (eg Ossenbruggen, 1994) see it as the product of probability and consequence. The consequence is usually but not necessarily expressed in monetary terms. The term *risk analysis* or *risk assessment* is used loosely to refer to several methodologies.

Risk assessment is an important part of the decision making process and management in general, and various methods have been developed to assist the process. The risk assessment process generally involves (1) identifying the failure modes or hazards in the system, (2) assessing the severity of those events, and (3) suggesting compensatory action. There are two ways of assessing risk, namely *qualitative* and *quantitative*, and these terms are widely used in the literature. The qualitative methods use textual descriptions of the risk (eg high risk ... low risk). The quantitative methods attempt to put numbers to the risk, usually by means of assigning a probability to the event (eg in units of failures per million operating cycles).

The *qualitative* risk analysis tools are used to identify what undesirable outcomes there may be in the operation of a system. For example *Fault Trees* may be used to identify potential failure modes. The benefit of the risk analysis is that these failure modes might otherwise have remained unanticipated. Once they have been identified, then it may be possible to design the faults out of the system or at least take precautions or put contingency measures in place. The qualitative methods stop once the risk modes have been identified: they do not determine the probability of occurrence.

In *quantitative risk assessment* (*QRA*) the emphasis is on quantifying the undesirable outcomes in terms of a probability. The primary usage is not so much to identify the failure modes and undesirable outcomes, but to assess the likelihood and consequences of uncertainty in a model. There have been many applications of QRA methods, typically financial. The interest is in determining not just the expected value of profitability in a new business venture, but also the probability of that outcome.

Probabilistic reasoning is a term which is has variable use in QRA. All the QRA methods analyse probabilities to a greater or lesser extent. Whether they actually apply a reasoning process to that analysis is more the question. The term is usually associated with Bayesian conditional probability, especially when coupled with an artificial intelligence application. A term such as *'probabilistic computation'* would possibly be more appropriate for the QRA systems that apply an algebraic operator to random variables.

Some QRA methods output a single point estimate of probability (eg Decision trees), while other methods produce a complete probability distribution. The production of a probability distribution has greater strategic value in the decision making process, as it gives not just the probability of success, but also the probability of any outcome that the decision maker may be interested in. Risk analysis is a management tool in that it helps identify or even quantify undesirable outcomes. The aim is to give better understanding of the possible outcomes, both those that are desirable and those that are not. In as much as to make a decision is to manage, risk analysis is therefore also a *management* tool. Of itself risk analysis does not make a decision, neither does Decision analysis. They only go as far as to provide information on which a rational decision may be made. Only *artificial intelligence* (*AI*) systems seek to automatically make decisions.

2.5.5 **Identifying and managing risk in the design process**

The uncertainties that exist at early design increase the risk that the design intent may not be realised. Consequently risk analysis methods have been applied to various aspects of the design process. For example

- Failure mode effect and criticality analysis (*FMECA*) (Ashley, 1993), hazard analysis and *reliability* assessment (Petersen et al, 1995[60]; Mazzuchi and Soyer, 1992[61]).
- Project scheduling risk assessment (Finley and Fisher, 1994).
- Health and environmental risk at chemically contaminated sites using probabilistic risk assessment (Kangas, 1996), *Life cycle assessment* (LCA) for evaluating environment effects (Keoleian & Glantschnig, 1994)[62].
- Testing (Bach, 1999)[63]
- Software risk (Chittister and Haimes, 1994)[64]
- Financial parameters such as cost and time to market using Monte Carlo (Kostetsky, 1994).

It has also been proposed that the whole product development process should be managed using risk approaches (eg Garcia (1994), Mar (1991), Pittman (1996), Ridgman (1996)). Sometimes this has been achieved, at least for specific domains. For example Bernhardt and Wolverton (1996) introduce a commercial software utility for managing the development of injection molded plastics parts, by assessing benefits and characterising the risks. Hyatt and Rosenberg (1998) developed a software program to "assess risk areas at each phase of the development *life cycle* and project them into the future", using measurable attributes to risks.

---

[60] Petersen et al (1995) state: “A great need exists to include reliability aspects at the early design stage to avoid non-appropriate design developments”. They used *functional modelling* for hazard analysis and *reliability* assessment of systems when only the "operational requirements".

[61] Mazzuchi and Soyer (1992) discuss the need to assess system reliability during development, and the difficulty of doing this with sparse test results. They motivate for the use of Bayesian methods to combine subjective information with available test data.

[62] The life cycle includes the environmental effects of “raw materials acquisition, bulk and engineered materials processing, manufacturing, use, service, retirement, resource recovery and disposal”, (Keoleian & Glantschnig 1994).

[63] (Bach, 1999) describes testing as a "process of developing an assessment of product quality", and notes importance of managing risk and quality.

[64] Chittister and Haimes (1994) address modelling and management of software risk. They discuss the shift of importance away from hardware to software, and with it the necessity to manage software risk. They use their "hierarchical holographic modelling framework" to assess and manage these risks.

Crossland et al (1995) propose modelling uncertainty in the early design stages of design using an object-oriented model. Their intent is that both the level of detail and the level of certainty can be changed as the design progresses, in order to "represent uncertain relationships between objects thus permitting modelling of alternative (parallel) design paths, as well as uncertain attribute values".

While there are various risk assessment tools covering various aspects of risk, they have limited ability to interface with each other and thus a universal risk assessment methodology is currently unavailable.

2.6 **Recording and retrieving design intent**

Design intent is the rationale behind the design, i.e. the functional purpose that the designer intended to be achieved in the part. Drawings form the final visible record of design, but do not document the process of creation or the reasoning. CAD systems are strong at recording geometric intent (shape, size, tolerance, surface texture). However they are inadequate for making the whole design intent clear.

Design intent is often compromised downstream, in stages such as manufacturing, since it is difficult to transmit (Candy et al, 1996). For each design issue there is ideally a need to keep track of its alternatives and the arguments for those alternatives. This is necessary to support the ability to retract earlier design decisions. Another motivating factor for the formal recording of design decisions are legal requirements such as the European CE certification, whereby the producer is obliged to show that *safety* hazards are minimised (Kersten, 1996).

Although Mar (1991) believes that system engineering “provides a traceable and verifiable record of all engineering decisions", the systematic design processes are only able to provide this traceable record when all the parameters are quantitative.

The standard mechanism for recording and retrieving design intent is the mind of the designer. However this can be risky, as memory can fail, or the person may be lost to the organisation. Such design information that may be more formally recorded is frequently in personal notebooks and other records that are unstructured and inaccessible to others (Kersten, 1996). There are systems that go some way to addressing these concerns. One termed *ICAD* is a design system that captures geometry and also non-geometric information (design intent, part interdependencies and product structure, manufacture, company practices)[65]. Candy et al (1996) maintain that systems such as ICAD require more complete problem definition in that the constraints and parameters must first be identified by the designer.

The functional simulation system called *Schemebuilder* (Bracewell et al, 1996a) provides an "audit trail" of the design selection (see previous discussion). Other work in this area includes that of Keat et al (1996)[66] and Banares et al (1996)[67]. *Grammars* have also been used both for modelling design intent (Finger and Dixon (1989b).

Most of the models of design intent are in the exploratory stages. There are widely differing objectives among the systems, and in some cases it is not always clear what the objectives are beyond simply capturing design intent. Some models aim to manage the design process, keeping record of design alternatives and decisions made. Other models have the potential to evaluate product functionality and

[65]To achieve this, the system uses a declarative language with which the user provides the information. These descriptive attributes are linked to the geometric objects in an object-oriented system. Relational database tools are provided to create and access catalogue type data. Basic geometric objects are included so that a geometric model may be constructed. Included with the system are modules for surface modelling, parametric solid modelling, and the production of drawings. Although not specifically stated as such, the system appears to be close to an expert system in structure and function. The software is supplied by Concentra (http://www.concentra.com).

[66]Keat et al (1996) apply artificial intelligence to architecture. Their work is towards capturing design intent, on the basis that drawings of geometry only capture the end result of the design process, and not the development of the design.

[67]Banares et al (1996) specifically address the issue of design intent, and implement a system that records design intent. The system makes it possible to find whether an issue has been resolved or not, study reasons for choices, and identify which parts must be re-designed when a constraint changes (backtracking).

performance, by extracting geometric reasoning from a semantic description of a part, but these developments are still in their infancy.

The challenge with design intent is to develop systems that can record the design decisions, the alternatives, and the reasons. Importantly, there also needs to be a retrieve function, ideally one that will of its own accord recognise that an element of the design intent is potentially about to be violated, and then retrieve and present to the user that original intent. Perhaps such a system might also be able to show the consequential effects that a design change may have elsewhere in the system. It is speculated that a parametric solid modeller CAD system, with the addition of semantics to provide a qualitative representation of function, may be a solution space to be fully explored.

2.7 **Discussion**

2.7.1 **Classifying design tools with the Generic Design Activity**

A classification tool called the Generic Design Activity has been defined, and various design tools have been interpreted in terms of it. A summary is shown in Figure 2.19, giving the positions where some of the mechanisms (tools) and constraints apply.

The figure illustrates several inventive and simulation-assessment mechanisms. The Generic Design Activity is a generic creative building block and though it may be positioned anywhere in the design project, it does not necessarily follow that all the entries are applicable at every stage. Most of these are more suitable to the later rather than the earlier stages of design as they require relatively complete problem definition. These and other attributes of early design are now discussed.

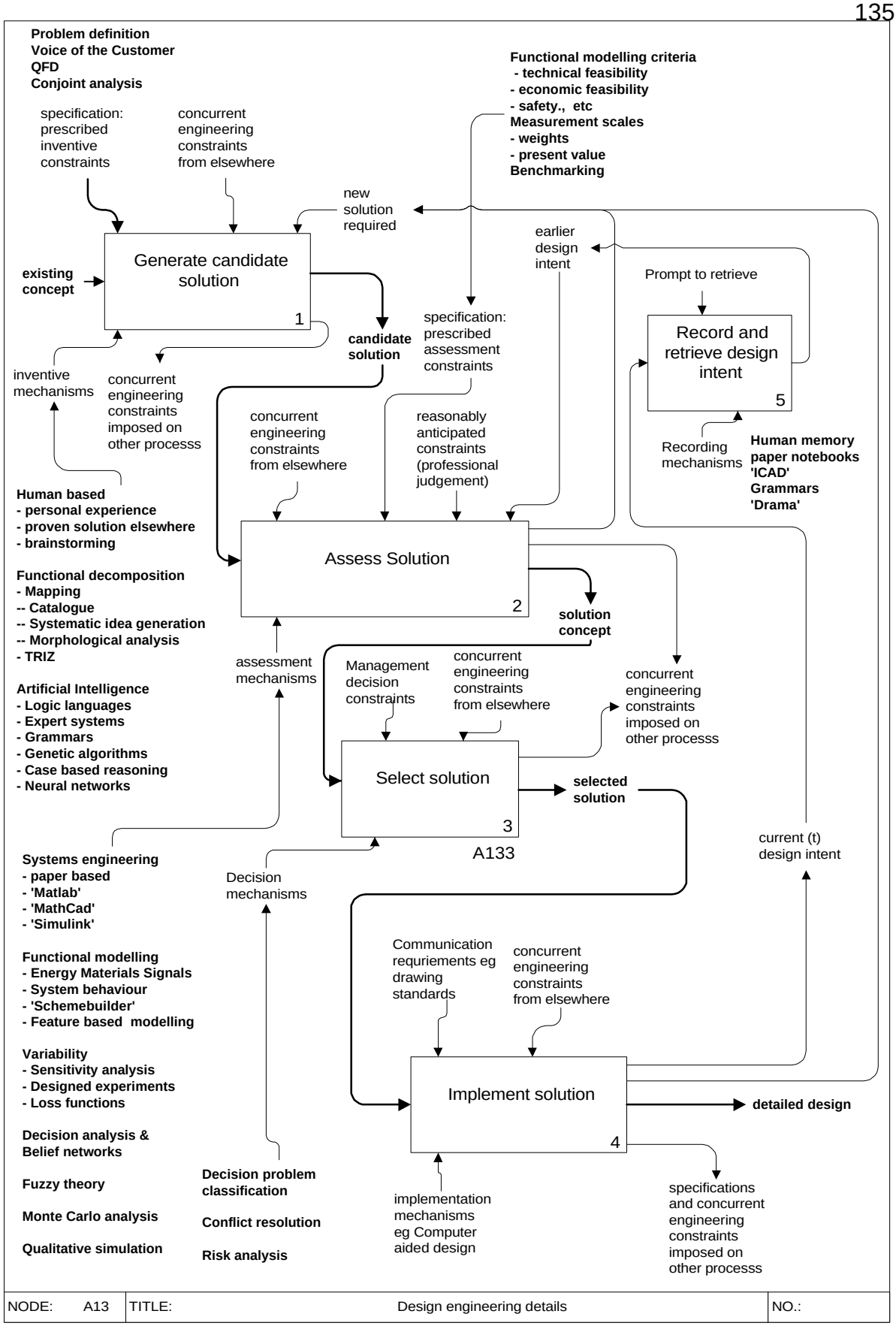


Figure 2:19: Generic Design Activity showing the positions of various tools and support systems in the design process.

2.7.2 Characteristics of Early design

The early stages of design are characterised by incompleteness, lack of confidence, and inconsistency of design knowledge (Ullman and D'Ambrosio, 1995). There is also an inability to apply formal procedures, and the risk of premature closure of concept design because of difficulty in generating and evaluating additional solutions (in turn due to the lack of support tools) (Candy et al, 1996).

Concept design is not simply the application of knowledge to a well defined problem. Instead there is a knowledge development that takes place in the designer's mind, and causes unforseen consequences in the process (Candy et al, 1996). One of the significant decisions that the design manager must make is how much effort is warranted to improve the design knowledge to such a point that the chances of a successful design are deemed adequately high.

There is a common refrain throughout the design literature of the need for design support tools to analyse designs at every stage from concept to detailed design (eg Finger & Dixon, 1989b). In particular Rabins et al (ASME Research, 1986) identify the need to understand the design process, particularly the early conceptual stages of design, and the feedback of *life cycle* design information. Following are some simulation and assessment issues that appear to be common concerns.

Simulation

- Need exists for models that can handle *qualitative* non-numeric parameters, sets of parameters (eg material choice).
- Explore how the design behaves, or how material properties affect behaviour (Finger & Dixon, 1989b).
- Show the *design intent*, i.e. how the design was intended to behave (Finger & Dixon, 1989b).
- Develop the ability to evaluate a configuration without being forced to assign values to the attributes, and identify the features on which to base such evaluation.

- Engineering analysis (eg FEA) requires complete geometry, which is not readily available at the early design stages. The use of a symbolic representation of the design is seen to be crucial for preliminary design analysis (Finger & Dixon, 1989b).
- Represent and evaluate tolerances in design, study the relationship between tolerance and cost, develop relationships between tolerance and function, account for tolerance stack up.

Assessment

- Measure and quantify life time performance of designs.
- Analysis and optimisation of design are often impeded by the lack of criteria (Finger & Dixon, 1989b).

2.7.3 Existing methodologies

The *functional modelling* systems are most closely aligned with the objective of simulating engineering performance, but the least able to accommodate uncertainty. Functional modelling appears to have the greatest prospects for assisting the design process. It leaves the creative aspects of concept formulation to the human designer, and avoids the issues of trying to generate solutions by artificial intelligence. Input-output transformations (using block diagrams and *graph theory*) provide a more quantitative output than the semantic approach, but of course this is at the expense of requiring more detailed information. They generally use block diagrams and *graph theory*.

The essence of the various functional modelling approaches is to produce a model of the solution concept, with known functional relationships between the components. In the case of the analytical behavioural models, these relationships need to be expressed mathematically, and they need to be available early in the concept design stages. In principle the method might provide a black box approach, in that the internal workings of a particular sub system need not be defined until such time as an

increase in resolution is required. *Semantics* (*grammars*) have attractive qualitative modelling capabilities that might make them suitable for the early design stages, but the methods are not yet robust and have been applied to relatively tightly focussed domains. The more analytical behavioural models have provided more tangible results.

There are some obstacles in the functional modelling approaches:

- *Distributed design* frequently occurs and it makes the *decomposition* process difficult. In turn the functional modelling and artificial intelligence methods depend on decomposition for success.
- The assignment of *functions* and their allocation to individual elements is easier in some fields (electronics, precision engineering, and power transmissions), but more difficult in general mechanical engineering (Pahl and Beitz, 1988).
- Functional modelling requires that relationships in the decomposed model be quantifiable. The technique runs into limitations when the relationships are qualitative, which is typical of the early design stages. It may be anticipated that it might not be straightforward achieving the required degree of precision for the mathematical relationships at such early stages, especially if the working principles are poorly understood. The idea of a progression from qualitative to quantitative analysis is mooted by Pahl and Beitz (1988) but not explored explicitly.
- The effort required to create the model and its relationships can be expected to be a significant human resource not to be taken lightly.
- Designers often have an initial function structure in mind when they start the design process. They might rarely start with a pure black box approach of inputs and outputs, (Schmidt and Cagan, 1993; Frost 1999).

Functional modelling systems may also be classified according to the level of *abstraction* involved. The early design stages deal with more abstract functions compared to the later detailed design stages (Schmidt and Cagan, 1993)[68]. The early stages of design are concerned with function, compared to the detailed design stages that are concerned with geometry (Deng et al, 1998). This parallels a progression from qualitative to quantitative knowledge during the design process. The modelling techniques differ according to whether the early or late design stages are being addressed. Other issues that arise when considering functional modelling are *determinism* (as opposed to variable parameters) and how *distributed design* (interdependency of function) is dealt with.

Artificial intelligence systems have sometimes been held up as a platform for developing solutions to the early design problems. Various artificial intelligence tools have been developed, but still the perfect artificial intelligence system for engineering design remains an elusive goal. The best achieved are partial solutions. Some of these are systems for small focussed domains, and others are generic systems that attempt to cover wide engineering domains. Arguably the better immediate successes have been achieved by concentrating on small well-defined domains. Moore and Miles (1996) discuss the merits of small versus comprehensive expert systems in concept design. They argue that large scale knowledge based systems have had poor success rates. Such systems are also difficult to develop and require substantial developmental resources. Conventional knowledge based systems are frequently too prescriptive, requiring rigid and limited user input. The poor acceptance of such systems in practical applications is further hindered by users resenting a sense of being replaced by an artificial system. Moore and Miles motivate for expert systems that are flexible, have graphical user interfaces, allow easy addition of new information, and assist designers in difficult areas rather than supplanting them in areas of strong human skills. They have thus concentrated on the development of

[68]Schmidt and Cagan (1993) view design as a process of starting in conceptual stages with the “generation and manipulation of abstract representations of functions and physical components”, and progressing along a continuum of decreasing abstraction until in the end it is defined in detail in terms of form. Different parts of the design may be at different stages of completeness.

multiple small systems that are focussed on small domains, focussed on concept design for civil engineering bridges. They describe four knowledge based systems in this domain[69].

However the smaller systems do not integrate well together. The big generic systems have the potential for the greatest rewards, if and when they can be realised into working systems for industrial use. However artificial intelligence systems are going to be limited in flexibility compared to a human, at least for the near future. Relatively few systems have progressed to the stage of robust prototype. Most of these have been evaluated informally, and practically no details are given about the implementation or the rules in the knowledge base. Also, it is readily apparent that expert system tools for engineering design suffer the same limitations as decision support systems (Ullman and D'Ambrosio, 1995), namely:

- Require complete information,
- Better at dealing with quantitative rather than qualitative information, and
- Deterministic rather than probabilistic (though they do handle logical functions).

Bartsch-Sporl and Bakhtari (1996) set out the requirements of artificial intelligence implementations as:

- Completeness of knowledge in the domain model,
- Consistent logic within the domain,
- Closed domain, not vulnerable to outside affects,
- Problems that can be decomposed and re-composed, and
- Solution spaces that can be formally defined.

They point out that real life design domains seldom meet these requirements precisely. This is confirmed by Tang (1996), who feels that artificial intelligence

[69]It is interesting to note that they are all developed in the C++ programming language, which is a procedural language rather than a conventional artificial intelligence language. System one monitors concept design only (no costing or sizing). It checks for violation of design constraints, and warns the designer. It suggests alternatives but does not prescribe the action to be taken. System two estimates bridge costs. System three advises on bridge aesthetics, by taking a proposed bridge and producing comments and a 2D drawing. System four is a proposed case based reasoning system for a subset within the bridge design domain. These systems are independent of each other, though the authors intend to link them together. The suite would operate like a blackboard, except that there is no central artificial intelligence manager. Instead the human user is expected to select which systems to execute.

design tools have only been successful in well defined applications that have fixed design strategies. Tang sees the central role of artificial intelligence in concept design as the provision of solutions, through proper representations and reasoning mechanisms, even though the design requirement is incomplete and inconsistent. Tang proposes an artificial intelligence system capable of deriving conceptual design solutions from functional design requirements.

Balance between human and artificial intelligence

Early strategies of artificial intelligence were perhaps over optimistic about what the technology could achieve. Now there is generally a more sober expectation of what artificial intelligence can and cannot do. It is realised that there are certain tasks, typically those that require judgement or subjectivity, that are better accomplished by humans than artificial means.

The strategy of Bartsch-Sporl and Bakhtari (1996) has been to use artificial intelligence, not so much to solve design problems on its own, but rather as an assistant to a human designer. Their approach is to keep the human designer fully responsible for the design and the decisions, while the artificial intelligence system helps present knowledge to the designer. The motivation for this approach is given as due to both acceptance and feasibility. Ansell and Mulhim (1993) describe the partial implementation of an expert system to analyse reliability data and assist the statistical analysis thereof. They too take a “non-authoritarian” approach to assisting the analyst, such that the user can chose whether the system performs basic calculations or assists with higher level strategic planning. The project appears to be incomplete, so its effectiveness is uncertain.

What does an expert system do well?

Procedural programming languages[70] are suitable for programs that can be modelled as a flow chart. Expert systems eliminate the need for flow charting. They still contain rules like procedural programming languages, but the inference engine in the expert system determines which rules are applicable in given circumstances, selects one of

[70]Example of procedural programming languages are C, C++, and Delphi among others.

these rules and executes it. The major advantage is that conditional statements can be avoided, and the developer need not determine or attempt to produce program code for all the situations for which a rule might need to be executed[71]. Consequently the order of the rules is irrelevant and new rules may be added without having to graft them into the existing rules. Haley[72] (1998) maintains that expert systems are suitable when one of the following applies:

- it is hard to produce a flow chart,
- there is a large amount of conditional logic,
- knowledge is neither strictly declarative nor algorithmic, or
- experience of special cases and exceptions is involved.

Production systems would appear to best meet the above requirements. The main contenders would appear to be CLIPS (NASA) and Eclipse (Haley).

Assessment by comparing requirements to simulated outcomes

A significant weakness of many functional modelling and artificial intelligence systems is the manner in which they assess the *fitness* of the design configuration. For example genetic algorithm methods require an assessment relationship to be defined by the designer, and this requires quantification of the problem, which may be difficult to do. Functional modelling systems are generally weak in addressing the assessment issues. This is partly because they do not directly need assessment, as they proceed from functional decomposition, which in turn originates in the specification. Assessment issues arise prominently in decision analysis, but they are generally solved using weights. This may be suitable for the financial domain where most such systems are implemented, inflation being an example of a weighted index. However the methods have weak rationales and better methods are necessary for the design domain where more complex utilities such as machine performance arise.

[71]However it should be noted that the procedural languages have very much better support for creating friendly user interfaces, whereas many of the expert systems, eg CLIPS, are rudimentary in their user interface and sometimes even use other applications as the interface.

[72]It may be prudent to observe that Haley is a commercial vendor for the ‘Eclipse’ expert system, so due allowance must be made for these statements.

Other intelligence tools

Expert systems provide knowledge analysis but little search capabilities, whereas genetic algorithms are the opposite, using search rather than knowledge. Expert systems and other artificial intelligence tools are essentially programming languages. They are specially strong at problem solving. It is interesting to note that some of the most impressive design support tools do not use artificial intelligence in any significant role (eg “Schemebuilder” as previously described).

2.7.4 Uncertainty at early design

Perhaps the characteristic that most clearly identifies the early concept design stages from later detailed design stages is the incompleteness and *abstraction* of information. Issues, constraints and criteria may be unresolved or weakly understood. Not all alternatives might have been developed to the same depth. It may not be possible to decompose the design problem into independent sub-problems. Constraints may be conflicting. They may also be qualitative[73], possibly with no prospects of ever being quantifiable. There may be undefined devices and unresolved functions. It is not always known whether desired technologies or alternative solutions exist. Proposed solutions in some areas may be incompatible with those elsewhere. Designs may have to change to compensate for solution constraints in other areas. Some decisions might still be open for technological reasons or risk aversion. Additionally, design is often undertaken by a team, where the individuals approach the problem from different viewpoints, knowledge and confidence.

This is a harsh environment for any support tool. It is no wonder that existing tools are limited in their abilities. However it is not only the artificial intelligence tools that struggle, since the early design stages are also difficult for human design managers.

[73]The interested reader is referred to Chapter 6 for a description of how the terms quantitative and qualitative are used in this work.

'*Uncertainty*' is the term loosely used in the literature to describe any or all of the above issues of incompleteness[74]. As this project is intimately involved with uncertainty, it is necessary to introduce the following clarifying terms:

- Uncertainty of analysis
- Variability of process
- Reliability

Uncertainty of analysis

The underlying mechanics of a design problem may be quantitative in a mathematical sense, in which case there is little uncertainty of analysis and functional modelling systems can operate satisfactorily. In other cases there are known boolean relationships, in which case artificial intelligence systems can usually operate but functional modelling begins to become difficult. If the problem formulation is qualitative then decision analysis methods may be appropriate, with functional modelling and artificial intelligence unable to operate easily.

Sometimes the uncertainty can be reduced by improving the resolution of the modelling process, for example finite element meshes can be refined to give improved accuracy of stress prediction. Improving the resolution works well when the underlying principles of the model are a true reflection of reality. In principle, and given sufficient computing resources, the resolution in such cases can be refined to such an extent that any residual uncertainty becomes trivial. However there are many problems that are not so well defined. For example the factors that affect wash performance of a dishwasher are not well understood. If there are relationships between wash performance and factors such as water volume and wash temperature, these are not sufficiently quantitative to construct a simulation model.

The fundamental limitation in such cases is the inadequacy of knowledge about the problem. The less that is known about how the system works, the greater the uncertainty of analysis. In the limit, when complete ignorance prevails, then it is not

[74]The concept of uncertainty having two forms is identified by Quelch and Cameron (1994), who describe the forms as "randomness inherent in the system" and "vagueness in the problem". Their work addresses the randomness aspect.

possible to make any prediction of outcome at all. Any prediction made under such circumstances would be totally unreliable. At the other extreme are cases that use laws of nature, for example the laws of celestial mechanics are well enough known to be able to predict the flight path of spacecraft to high accuracy. However, even laws of nature have an uncertainty, albeit small.

Variability of process

Another source of randomness in outcomes is *process variability*. By this is meant the randomness that comes from production and other processes. For example, the dimensions of a part will not be exactly the same for part after part, but will show variability. Changing or improving the fabrication process is an important part of reducing the process variability. It is necessary to collect data to quantify the variability, and hence the importance of process control and reliability measurement. Variability may then be measured in terms of a probability distribution. Elsewhere process variability has sometimes been termed *stochastic uncertainty* (Wood et al (1989).

Reliability

This factor refers to the probability of function being available at some time in the future. It is the variability of a functional attribute with time. A typical example is the fatigue life of a device. Both reliability and process variability are a probability distribution, but reliability refers to probability as a function of time, whereas process variability refers to probability as a function of a parameter such as dimension.

The process variability is comparatively easy to quantify, whereas the analysis uncertainty is often considerably difficult to establish. The process of combining uncertainty and variability to determine the total randomness is not well established.

2.7.5 Multiple viewpoints on design

A functional modelling system may be classified firstly by the *viewpoint* that it models, eg energy, material flow, control signal, geometry (tolerances), fabrication process, etc. The issue of viewpoint is fundamental to functional modelling, in that a model has to be constructed for the viewpoint concerned, eg the relationships for the energy viewpoint will not in general be the same as for the material handling domain. Most of the models that have been developed are restricted to one viewpoint. The common viewpoints in this regard are conservation of energy, and geometry (including relative location). There do not yet appear to be systems that can model a design from multiple viewpoints. However the need to do so has been clearly identified in the literature. For example Hague et al (1996) believe that the reluctance of designers to change their initial concepts may be due to lack of information on *life cycle* issues. They motivate for the development of methods to evaluate concept alternatives from multiple viewpoints (design, manufacturing, assembly). Likewise Rosen et al (1994) discuss "functionality" as a primary viewpoint, with the need to evaluate designs in the secondary viewpoints of "manufacturing, cost, and other life-cycle considerations". A framework for managing three perspectives of the risk in software development has been proposed by Chittister and Haimes (1993).[75] However on the whole there are few tools that actively support multiple viewpoints of design.

[75]Each of these perspectives are sub-models of the whole, and each perspective is broken down in an hierarchical manner. They term the proposed framework "hierarchical holographic modeling". The first perspective is functional decomposition, which in their context refers to "requirement, product, process, people, management, environment, and system development". The second perspective is "source-based decomposition", which covers failure sources namely "hardware, software organizational, and human". The third perspective is "temporal decomposition", which refers to the "stages in the software development process". At each level they seek to identify the risks, and for this they pose a battery of questions, eg "What can go wrong? What is the likelihood that it will go wrong?" among others. Once the problem areas are identified, then they rank the risks.

2.8 **Conclusions**

The required characteristics of a methodology for early design are:

1 The issue of *viewpoint* is important. There is a need for the designer to acknowledge the existence of viewpoints other than function, and actively design for them. It is desirable to have tools to assist this process. Viewpoints involve the capability to anticipate other views and see how a change in one area affects the system performance in another viewpoint.

2 Any modelling system for early design should be able to cope with is *uncertainty of analysis (incompleteness of knowledge).* Ideally the relationship by which input variables determine the outcome will be a mathematical expression. However this degree of knowledge is not always available, especially for design projects that break new ground, as there may be incomplete knowledge of system behaviour. Instead the available knowledge may be logical rules, or even only (expert) opinion. The problem of low uncertainty of analysis also arises when input variables are nominal, since this invalidates any mathematical expression that might otherwise be relevant. The uncertainty of analysis is a characteristic of early design that is particularly problematic for existing design methodologies.

3 *Process variability* need to be modelled, which is to state that information is not always deterministic but may be probabilistic (eg the dimensions of a part off a production line, or the failure states of a machine system).

4 It is necessary to accommodate varying degrees of *information abstraction*, both quantitative variables (ratio and interval scales) and qualitative variables (ordinal and nominal scales), and their mixture. Both types of variable may have process variability, and there may be different degrees of knowledge

about how the variables are related. Methods for performing probabilistic computation on both quantitative and qualitative parameters are required.

Most of the inventive mechanisms and simulation-assessment mechanisms are more appropriate at later design stages as they need more complete information. The methodologies and tools that aim to *support* rather than supplant the human designer are probably more successful. Artificial intelligence methods seek to create design solutions, but this strategy has had limited success except in well defined domains. There is a need for methodologies that meet the above list of characteristics at early design.

Chapter 3

Project strategy

The previous chapter reviewed the literature on simulation systems that might be usable at early design. This chapter describes the intent and strategy of this project.

3.1 **Supporting early design**

There is wide acknowledgement that high *uncertainties* exist at the early design stages, that this is a major problem both for human designers and their tools, and that the design decisions made at the early stages have significant downstream consequences, in terms of a number of *life cycle* issues such as function, reliability, manufacturability, etc. However the solutions that have been proposed have been widely divergent in what they intended to achieve. Perhaps the single greatest differentiating factor is *solution generation* versus *system modelling*.

The *expert system* and *artificial intelligence* methods attempt to generate solutions. It would seem that implicit in this approach is the belief in an *ignorant designer*: that the designer might be unaware of potential solutions, or indeed may be a total novice. Accordingly the expert systems and neural networks steer the design down paths which have in the past proved successful. The genetic algorithms take a slightly different approach in that they decompose the design problem, apply a combinatorial synthesis of solutions, and then weed out the underperforming designs.

The alternative approach to early design is to support the decision making process, and not to generate solutions. The *functional modelling* systems fit into this category. However the difficulty with modelling systems has been the lack of tools to incorporate (i) process variability and (ii) uncertainty of analysis (i.e. qualitative relationships) into the analysis alongside crisp quantitative facts. Whether the solution generation or decision support direction is better is debatable. There have been some perhaps spectacular solution generation tools, but their success has generally been in very narrowly defined domains.

The decision was made in this project to avoid the concept generation route, and instead concentrate on supporting early design by developing a simulation system that could operate on both quantitative and qualitative[76] data. The functional

[76]The interested reader is referred to Chapter 6 for a description of how the terms quantitative and qualitative are used in this work.

modelling and artificial intelligence methods have too much difficulty with uncertainty to be immediately useful. The decision analysis tools have promise, but do not process quantitative mathematical relationships. It was therefore necessary to develop an alternative methodology.

3.2 **Statement of hypotheses**

Given the existing literature as described in the previous chapters, and the limitations of existing methodologies to support the design activity, the project objective was as follows.

The hypothesis of this project is that a methodology can be developed for simulating and assessing design integrity at early stages, processing both qualitative and quantitative information, accommodating uncertainty of analysis and process variability, and modelling the performance of the system from multiple viewpoints.

The objective was therefore to develop a method whereby issues of design *integrity* may be evaluated during the early stages of say mechanical product design. During these stages a design is characterised by large uncertainty, and design concepts are unsettled and amorphous. The project has the potential to provide methods to help *manage* the process of product design. The conceptual stages of a design are critical to the long term technical success of the machine. It is during concept design that various choices are made: some technologies are included and others are excluded, and the consequences carry through to subsequent design stages. The cost of revising the fundamental concept becomes an increasingly prohibitive barrier the further downstream the design moves.

A successful project would address issues such as:

Simulate design alternatives

Permit alternative solutions or physical devices to be evaluated, even when the precise performance of those devices is unknown.

Predict downstream consequences

Help evaluate design alternatives during the early design stages, by raising issues that the designer may not have considered. Address issues such as probability of the design intent being met (robustness), life of product (reliability) and life cycle costs to the producer and the consumer.

Optimise the product

Assess the effect of a design change in one local part of the machine on other parts. Reduce the time needed to explore and evaluate design alternatives.

3.3 **Solution approach**

This project develops a simulation methodology that permits uncertainty to be modelled systematically, so that design *integrity* may be ensured from the early stages. ‘Integrity’ is used to refer to the full and multifaceted integrity of the product being designed. In this context a design is considered to have integrity if the product robustly meets all the requirements of the various stakeholders.

In the case of a dishwasher the various stakeholders would include the Producer, the Consumer, and Society in general, and between them they would have requirements for profitability, wash performance, noise, water usage, power consumption, safety, liability, environmental friendliness, and possibly others. Importantly, they would require these product characteristics to be *robust* against perturbations. For example, wash performance should be close to that predicted at early design (i.e. low

uncertainty of analysis), consistent from product to product (i.e. low process variability) and also reliable over time.

Realistically, there is no guarantee that a design solution can be found which satisfies all requirements. This is increasingly so with complex machines designed by multi-disciplinary development teams and with many stakeholders. Although it may be impossible to perfect a design that meets all requirements, the designers and their managers can still optimise it to at least partially meet some of the requirements. Multiple solutions will then be possible, depending on which viewpoints are given priority in the optimisation process. It is therefore imperative that designers are able during optimisation to take in the viewpoints from the broader picture, and not only that part of the design problem with which they are involved. For example it is not uncommon to encounter consumer products that provide the required function when new, but which have poor reliability over time. Some of these reliability issues could have been prevented at negligible cost had the designer been able to perceive the viewpoint beforehand. It is therefore necessary for an integrity assessment tool to be able to accommodate multiple viewpoints. Furthermore, each viewpoint needs to be able to accommodate and display uncertainty, so that a decision may be made with knowledge of the risks and how the unknowns could affect the outcome.

Optimisation strategies usually attempt to convert all viewpoints into the same units, typically financial value, for comparison purposes. The difficulty with optimising a design is that the different viewpoints (eg noise, wash performance, and cost) cannot always be converted to the same scale. Sometimes the units of measure may be different, or else they may be entirely qualitative (eg attractiveness of styling would be on a qualitative scale). Optimisation methods attempt to overcome these problems by assigning *weighted* scores to qualitative values, and then seeking a design where the total score is optimised. However this can be a contentious approach, as the scores are vulnerable to subjective bias, qualitative scales do not necessarily convert to linear numerical scales, nuances may be overlooked, and the optimisation algorithm can be difficult to justify.

This project avoids the weighted scale controversies and makes no attempt to optimise a design solution automatically. Analogy with fielded expert systems and other artificial intelligence applications would suggest that tools that make the decision for the designer require for success to be tightly focussed on a particular domain, and need stable and complete problem definition (Bartsch-Sporl and Bakhtari, 1995). Instead this project aimed to develop a generic design support methodology that could present information so that a human user can make an informed decision.

3.4 **Development milestones**

3.4.1 **Initial approach**

The original objective was to develop a system that could predict machine performance at the early design stages, eg wash performance of a dishwasher. The intended approach was to develop a quantitative model. To do this required investigation into the operating mechanics of the product, and the development of a mathematical model. The model might then have been interrogated to perform optimisations and what-if studies. In particular there was a desire to be able to change one mechanical subsystem (eg a pump) and see the effect on the rest of the system. Effectively this was a *functional modelling* problem. In addition there was the desire to be able to explore the design from multiple viewpoints such as performance and cost.

Initially the work progressed along functional modelling lines. However the existing functional modelling methods do not accommodate the uncertainties of early design very easily. Consequently the *artificial intelligence* methods were investigated. The thought was that although less precise than mathematical modelling, if the relationships describing performance could be expressed as logical rules then an artificial intelligence method such as an *expert system* could be developed. It soon became apparent that the artificial intelligence methods also do not accommodate

uncertainty well, and need to be constrained to well-defined domains. While *decision theory* does process uncertainty and is not domain specific, its drawback is that it does not process quantitative relationships such as mathematical or boolean statements, and therefore has only limited usefulness in the intended modelling domain.

Investigation has shown that there is no body of public literature on dishwasher wash performance. The principles of wash action have not been written about, if they have been researched at all, and there are no models of system performance in existence. There are many manufacturers of dishwashers and it is possible that they have confidential research findings which they keep to themselves. However even this could be a risky assumption given the remarkable similarity of design across different manufacturers. The available information on wash performance is therefore limited to qualitative expert opinion. This makes very difficult terrain for functional modelling. Worse, with the early design being a time when design parameters are highly uncertain, it is not clear how uncertainty could be processed through a model that was of itself highly qualitative, even assuming such a model could be created.

3.4.2 **DSI development**

The project needed to have a tool that could process quantitative and qualitative variables, with their process variability through relationships that were uncertain. Decision theory could accommodate relationships based on opinion, and *Monte Carlo* analysis the mathematical relationships, but the literature was silent on the two working together.

The constraint is that Monte Carlo is usually used with a mathematically explicit inverse function, with use of a histogram as the driving distribution being relatively rare. More critically, the Monte Carlo method cannot cope with a qualitative scale (eg ‘good..bad’). The ability to process a histogram through a quantitative probabilistic

computation system is necessary for an integration of quantitative and qualitative methods, since decision tables can only produce histograms.

The solution was to develop a methodology with which to investigate the main thesis of this project. This needed to provide both qualitative and quantitative probabilistic computation.

The methodology developed here has been called *Design Integrity* or *Design for System Integrity* (DSI). It was developed into a software embodiment to demonstrate its validity. DSI is computationally demanding, so software help is required to make it practical.

3.4.3 **Related developments**

In the testing and benchmarking process after the DSI software tool had been developed, it was discovered that related, though not identical functionality had been developed commercially in the form of 'Analytica' software.[77] The similarity of function, specifically the ability to process both qualitative and quantitative data, precludes this project from claiming novelty in that regard even though the developments were independent. However while the functionality may be similar, there are significant differences in the internal algorithms that the two methods use. The DSI solution uses a new algorithm for quantitative probabilistic computation using a combinatorial process. This substitutes for the functionality of the Monte Carlo algorithm and integrates easier with decision theory as both use histograms. Importantly, the use of this algorithm means that DSI does not have the random modelling artefacts and the solution convergence issues of Monte Carlo (and hence Analytica too), and these are potentially significant user benefits. No record could be found in the literature of prior description of this algorithm, nor does commercially

[77] 'Analytica' version 2.0 for Windows was tested. It uses Monte Carlo analysis for the quantitative probabilistic computation, and a decision table for the qualitative computation. The software is produced by Lumina Decision Systems, 59 N. Santa Cruz Avenue, Suite Q, Los Gatos, CA 95030. Web site contains more details at http://www.lumina.com

available Monte Carlo analysis use it, and it may be novel. However the thesis of this project does not rest on the novelty or otherwise of the computational algorithms, as these are only the tools to explore other objectives. Issues of the tool aside, there is no record in the literature of any qualitative and quantitative tool, including Analytica, being used to explore the early engineering design stages.

3.4.4 **Other features of DSI**

In principle it is possible to input histograms into Monte Carlo, and a suitable algorithm was also developed and demonstrated in this project (for comparative purposes). However the combinatorial joint probability algorithm is faster and more accurately shows distribution shape, and is therefore the application of choice. A Fuzzy theory cut-set algorithm was also provided in DSI. The relative merits of the various algorithms are discussed in Chapter 4.

The multiple viewpoint objective was also met in the DSI software, with the capability to have multiple views open at once, and sharing data. Also, semi-automatic creation of auxiliary views is provided for in the software, for example the background creation of elements of a reliability view while working on a cost view. Support was also provided for one device (eg a pump) to be substituted with a new type of device such as a different brand, and have all the related properties change in all open views. These capabilities differentiate the current project from all prior existing methodologies, Analytica included. Further details are provided in following chapters.

The DSI system can process both qualitative and quantitative relationships, and can operate at early design where information is sparse. It incorporates the functionality of Monte Carlo and decision analysis, and can therefore be used wherever those methodologies are applied for risk analysis. It is therefore also able to provide *quantitative risk assessment* (*QRA*). It may be used to perform probabilistic

computation on fault trees. It may be used to construct probabilistic models that predict machine performance metrics at the early design stages.

DSI is a *functional modelling* and simulation system that adds the probabilistic dimension to the simulation. It accommodates uncertainty of analysis as well as process variability. Once created a model may be explored by changing a variable (eg pump pressure) and propagating it to see the effect of the change. Multiple viewpoints are possible, with data being shared by viewpoints if required. It is not an automatic designer that makes decisions or searches for solutions (cf *genetic algorithm*).

The central example used for illustration in the project is the domain of domestic automatic dishwashers, for which integrity might be evaluated from multiple viewpoints such as wash performance, energy consumption, noise, ease of manufacture, cost, safety and reliability. A key feature of the DSI methodology is its ability to propagate (at concept, embodiment or detail design stage) both qualitative and quantitative random variables through relationships that may be uncertain.

3.5 **Application of the DSI methodology**

The approach in this project was to rely on the human designer for the creative design skills, as well as the assessment of the design, and have the methodology assist the decision making process and the management of design.

Each subsystem or device in the model will have properties (e.g. function, cost) with relationships linking them together. At concept design the primary objective is to determine form and function, but as the design matures so specification of detail (e.g. drawings) and design for manufacture become important. Throughout the design process the designer needs needs to understand how the relationships between devices affect the overall integration of the design, and ultimately the cost, performance, robustness, reliability and other key characteristics of the system.

Many of these relationships are intrinsically uncertain, as the system behaviour may be incompletely known.

In use the DSI process is initiated by an expert, probably the designer, who identifies the key devices in the model. In the case of the dishwasher the system may be defined with physical devices, eg "wash pump", "sprayer", etc. Each of these is attributed properties that relate to one or more of the viewpoints from which system integrity is to be evaluated, such as "wash pump cost", "wash pump pressure", and relationships are defined for each of the multiple system integrity viewpoints. These relationships are different for each of the viewpoints under scrutiny, and can be mathematical (e.g. addition, subtraction, multiplication, and division), logical (e.g. maximum, minimum), or opinion (decision table with uncertainty).

The probabilistic approach means that any input variable can be accommodated, whether ratio or nominal scale. The DSI method does not require an order in the input variable as does Fuzzy theory, nor are weights or scores applied as in *QFD*. The method also permits any combination of quantitative and qualitative variables to exist in the same model.

When there is no uncertainty of analysis, i.e. a mathematical expression fully defines the outcome in terms of input variables), then DSI still retains the ability for those inputs to have process variability.

Once the system has been defined by appropriate relationships, the end user (who may be different from the expert), determines the random variability in the inputs. For example, the user might assert that water temperature is certainly ‘warm’. This can then be propagated through the system to determine the resulting wash performance and other outputs. A more realistic assertion at early design stages might be to give a probability distribution, e.g. 30% hot, 50% warm, 20% cool, 0% cold, to encompass the uncertainty that exists at this stage. That is, the designer is leaning towards using a hot-warm set point on the thermostat, but wants to include the smaller possibility of a cool running option.

Once distributions are specified for all the inputs, the outcomes are computed as probability distributions. If necessary, alarms (acceptance limits) can be defined by the user for any parameter, and these can be checked during computation.

The ability to propagate both qualitative and quantitative variables through a probabilistic computation despite uncertainty of analysis, means that:

(i) the methodology can be applied to a number of different domains, and in particular, may be used in the early design stages when data are typically sparse,

(ii) system integrity may be assessed in a way that is impossible using conventional deterministic methods,

- the ability of the methodology to propagate the effects of a change in a design parameter means that the "risk" in the result (e.g. total product cost) may be identified, quantifying how good or bad the outcome may be.
- Effects from various viewpoints may be studied concurrently.

The computational methods are described in detail in following chapters. Briefly, the process involves taking two input probability distributions, applying a mathematical operator (such as +, -, x, /, maximum, minimum) or a decision table (map) to determine an output distribution.

In the early stages of design, the probability distributions associated with key characteristic should be expected to be broad, indicating the significant uncertainties in analysis and process variability at this stage. These uncertainties should reduce as the design progresses to more concrete stages. Variables that were qualitative may become quantitative, and the probability distributions narrow as process variability decreases. Also, new knowledge may be discovered, e.g. from prototype test results, so that relationships based on opinion may be replaced by mathematical expressions, i.e. reducing the uncertainty of analysis. Reduction of process variability and uncertainty of analysis both result in narrower probability distributions of key characteristics, and a corresponding clarity of risk and design integrity.

It is to be expected that residual uncertainty will always remain. In many cases the effort required to convert a subjective relationship into a mathematical one may not be worth the effort as it involves researching new knowledge. Therefore uncertainty of analysis may well exist even at advanced stages of design. Process variability exists at all stages, even into production, where it is linked to quality issues.

By inference, the methodology is able to identify sensitivity of the system design to particular parameters and how tightly tolerances in any given parameter must be held to achieve a robust design. As the methodology produces a probability distribution for each parameter in the model, this distribution may be used in design management, particularly to make decisions regarding viability and risk.

Unlike expert system tools where the software seeks to make a decision using artificial intelligence method, the DSI method does not make decisions itself. The method is an analytical assessment tool which supports the human decision making process by providing information about risk probability. Unless numerical risk acceptance criteria are incorporated in the model, it is up to the human manager to make the decisions, based on his/her own tolerance of or aversion to risk.

3.6 **Conclusions**

The move to concurrent engineering process places pressure on the design function. The development process is compressed and production process are typically initiated before the design has been fully built and tested. Therefore it is necessary that the integrity of a design be evaluated earlier in the design process so that problems can be addressed. Designers can have difficulty simultaneously grasping the multiple relationships between systems, subsystems and components, especially when multiple viewpoints (eg function, reliability, cost etc.) are necessary. This chapter has outlined a DSI methodology, which has been developed as a software tool, to assist in faster evaluation of system integrity from multiple viewpoints, even at early conceptual stages. The methodology fuses quantitative and qualitative

assessment, and incorporates probabilistic computation. Implementation of the method, in its quantitative and qualitative aspects and its multi-viewpoint capability, are illustrated in more detail in following chapters.

Chapter 4

Quantitative probabilistic computation

The development of an alternative numerical probabilistic method to Monte Carlo analysis is described. The methodology combines two quantitative random variables at a time with a mathematical operator, and produces an output distribution. In principle any type of probability distribution or histogram may be modelled, and different distributions may be used for the two inputs. The method has been implemented in software as it is computationally demanding. A variety of common mathematical operators have been demonstrated, and the principles are extendable to other operators. The existing suit of operators may be used to model various relationships such as function, performance, reliability, dimensional tolerances, cost, etc., in the quantitative domain.

4.1 Introduction

The primary purpose of an analysis or simulation system is to determine the output of a system. Most engineering analysis achieves a deterministic output. Likewise conventional *functional modelling* work is generally based on single point values, which are propagated through the simulation. Such approaches give no indication of the probability of the simulated outcomes, and there is no way to check the risk or integrity of the design. There is widespread anecdotal support for the belief that identifying and resolving design risk at the early design stages rather than during production decreases the product costs and the impact on the development programme (eg Belev, 1992). This chapter summarises the available tools for quantitative probabilistic computation, and then describes a new different approach.

The quantitative assessment of risk necessarily involves the quantitative analysis of probability. However there are different degrees of complexity of process and these will be now be summarised.

4.1.1 Interval analysis

The simplest form of scenario analysis is to represent a parameter as an interval, and then repeat the simulation with the parameter variously at either end of the interval (*what-if analysis*) or in systematic combination with other variables (*sensitivity analysis* and *tornado diagram*). A formal mathematics of interval analysis is also available. Limitations of the methods are:

(1) They show the range of outcomes, but do not quantify the probability of those outcomes.

(2) The interval for an input variable is difficult to interpret consistently: does it represent the commonly encountered range, or the likely range, or the maximum range conceivable?

(3) The output range is excessively conservative. This is because it assumes that all parameters are simultaneously at either the most or least favourable

position in their range, which becomes increasingly unlikely as the number of independent variables increases.

The interval methods provide a quick and simple mechanism to determine the approximate range of the output. They can provide the useful function of indicating which inputs contribute most to the uncertainty of the output. However the methods are conservative in representing risk, and probability has to be included if it is desired to improve the resolution of the analysis.

4.1.2 Single point probability variables

In the simpler probability analysis systems the probability is represented by a few discrete values, usually a binary set such as probability of failure vs that of non-failure. Such systems include fault tree analysis (FTA), failure modes and effect analysis (FMEA) and *decision theory* (including decision trees and influence diagrams, though these may have more than two discrete states).

Qualitative parameters[78] (eg 'failure, no failure') are also possible with these systems, with a numerical probability given to each state. Somewhat confusingly, these methods are nonetheless referred to as *Quantitative risk assessment* (QRA) as they quantify probabilities even if the states are qualitative. Most of the quantitative risk assessment methods produce a single point probability value for the outcome of interest, eg., there is y percent probability that the system will fail, and (100-y)% probability of it succeeding. Decision analysis also produces a probability value against a qualitative outcome.

[78]The interested reader is referred to Chapter 6 for a description of how the terms quantitative and qualitative are used in this work.

4.1.3 Special methods for combining certain random variables

More powerful mechanisms are available to determine outcomes as complete probability distributions. The input variables are necessarily quantitative. The difficulty is processing the probability distributions through conventional mathematical operators. An analytical method exists for addition and subtraction of Normal distributions. This requires only the mean and standard deviation of the inputs to determine the mean and standard deviation of the output[79]. However the method is limited to the Normal distribution and to addition and subtraction.

In the engineering design domain this method forms the basis for analysis of *load-capability interference*. It has applications in other domains, for example Sarper (1994) used it for *capital rationing*, which is to select projects out of a collection of candidate projects in such a way as to maximise the financial return. Sarper used *net present value* (NPV) as the criterion, with inputs represented by probability distributions. Sarper would have liked to have used uniform random variables, for reasons which are not apparent. However Sarper found that the methods of algebraic manipulation on uniform distributions were not as well developed as for the Normal distribution, and therefore converted all inputs to Normal distributions so as to use the Normal methods. Whether the convenience justifies the sacrifice in accuracy is debatable. However it does indicate the scarcity of methods to combine random variables.

Other methods, some of them approximations, are likewise known for other distributions and mathematical operators. For example, Rai & Krewski (1998) describe the development of a model for multiplicative risk, that is the product of two or more risk factors. Again, *reliability* analysis typically assumes a constant *failure rate* (eg. units of failures per million operating cycles) regardless of system age. This

[79] Given input X_1 with mean μ_1 and standard deviation σ_1, and similarly a second input X_2 with μ_2 and σ_2, then a net worth given as the weighted sum $Y = a_1.X_1 + a_2.X_2$ (where a_1 and a_2 are constants) will have mean $\mu_Y = a_1.\mu_1 + a_2.\mu_2$ and variance $\sigma_Y^2 = a_1^2.\sigma_1^2 + a_2^2.\sigma_2^2 + 2.a_1.a_2.\sigma_1.\sigma_2.\rho_{12}$ where ρ_{12} is the correlation coefficient (which is zero if X_1 and X_2 are independent).

equates to the *exponential* failure distribution, and makes it possible to apply tools (including as *Markov* analysis) to mathematically process “OR” as well as “AND” logic, plus complex interactions such as redundant and voting systems. Queueing theory uses similar mathematical tools and likewise relies heavily on the exponential distribution.

However only a special few distributions and operators have a known method, in which case it is necessary to resort to tools with greater power, as described next.

4.1.4 Algebra of random variables

The *algebra of random variables* (sometimes called the *probability calculus*) is a mathematical solution to the problem of combining two input *random variables* (each represented by a probability density function) using an algebraic operation (eg plus, product etc.). Methods of determining the sum of independent random variables have received much attention, but not products and quotients (Springer, 1979). A summarised historical perspective is provided by Springer.

Given variables x and y from input densities $f_1(t)$ and $f_2(t)$ respectively, then there are a number of approaches:

(1) Integration of joint density
In the case that the *joint density* f(x,y) is known, then the probability of the outcome (Syski, 1989) is

$$pr(A) = \int\int_{R(x,y)} f(x,y)\ dx.dy$$

where
R(x,y) region in the x-y plane, which is determined by the nature of the algebraic operator

Unfortunately the joint density is seldom available, and one of the following methods has to be used instead.

(2) Convolution of densities

The method involves a convolution of the input densities (Springer, 1979; Syski, 1989). For example, if the inputs are independent and for an operator of w = x + y, then the cumulative probability G(w) is given by[80]:

$$\begin{aligned} G(w) &= pr(x+y \le w) \\ &= \iint_{x+y \le w} f_1(x).f_2(y)\ dx.dy \\ &= \int_0^w f_1(x)\ dx\ .\ \int_0^{w-x} f_2(y)\ dy \end{aligned}$$

and then the density of w is g(w) is the derivative of the above and is given by the following convolution:

$$g(w) = \int_0^w f_1(x).f_2(w-x)\ dx$$

This convolution process is different for each algebraic operator so cannot be given in general. Having established the convolution integral, the next task is to substitute the functions for the two input densities and seek to complete the integral.

For example, if the inputs are both uniform distributions over the interval 0 to 1 ($f_i(x_i)$ = 1, $0 \le x_i \le 1$, i=1,2) then the sum convolution is (Springer, 1979):

$$\begin{aligned} g(w) &= \int_{range\ of\ x_2} f_1(w-x_2).f_2(x_2)\ dx_2 \\ &= \int_0^w dx_2 = w,\ for\ 0 \le w \le 1 \\ &= \int_{w-1}^1 dx_2 = 2-w,\ for\ 1 \le w \le 2 \end{aligned}$$

[80]Note y = w - x

This is the probability density of the output, and it is a triangular function [81] triang(0,1,2).

The product $W = X_1 + X_2$ of two random variables (Springer, 1979) is

$$g(w) = \int_0^\infty \frac{1}{x_2}.\ f_1(w/x_2).\ f_2(x_2)\ dx$$

However convolution can be an involved task as the density functions are not in general the same and integrate with various degrees of ease. When multiple random variables have to be operated on then the result from one convolution is input into another, and Syski (1989, p44) reports that 'unfortunately such calculations are very tedious and in most cases prohibitive, even in the simplifying situation when all life times x have the same density'.[82] Quotients and products are yet more troublesome.

(3) Transforms

Sometimes it is easier to apply a transform to the problem, and then an inverse transform (Springer, 1979; Syski, 1989). Thus, if X_1, X_2, ...X_n are independent random variables with densities $f_1(x_1)$, $f_2(x_2)$ etc, then the probability density of the sum of those random variables is given by the inverse Fourier transform of the product of the Fourier transforms of $f_1(x_1)$, $f_2(x_2)$ etc, namely:

$$g(w) = F_t^{-1}[\ \Pi_{j=1}^n\ F_t(f_j(x_j))\ ]$$

$$= \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{itw}\ \Pi_{j=1}^n\ F_t(f_j(x_j))$$

[81] Both Design for System Integrity (DSI) and Monte Carlo produce this result too, though they use numerical methods instead.

[82] A common application is the sum of several random variables representing cost. If these are all uniform distributions, then the convolution of the first two distributions yields a triangular distribution. Unfortunately the convolution of this triangular distribution with the next uniform input produces no simple distribution shape. The mathematics becomes more involved with each such step.

where F_t refers to the Fourier transform, and F_t^{-1} the inverse transform (Springer, 1979). The difficulty with this approach is performing the inverse transform as it may be cumbersome (Syski, 1989). The transform approach works best for independent random variables that are identical and of singly infinite range (0 to ∞) (Springer, 1979, p64). When the variables have finite or double infinite ranges (eg Normal distribution) then the sum has to be partitioned into separate ranges.

Depending on the algebraic operator it may be necessary to use either the Laplace, Fourier or Mellin transforms (Springer, 1979). The Laplace transform of function f(x) and the inverse Laplace transform are (respectively):

$$L_r(\ f(x)\ ) = \int_0^{\infty} e^{-rx}\ f(x)\ dx$$

$$f(x) = \frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} e^{rx}.\ L_r(f(x))\ dr$$

where r is a complex variable ($r = x+iy = R\cos\theta + iR\sin\theta = Re^{i\theta}$ where $R = (x^2+y^2)^{0.5}$ and $\theta = \arctan(y/x)$ and x and y are real numbers).

The Fourier transform and inverse Fourier transform are (respectively):

$$F_t(\ f(x)\ ) = \int_{-\infty}^{\infty} e^{itx}.\ f(x)\ dx$$

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-itx}.\ F_t(f(x))\ dt$$

The Fourier transform represents an ‘arbitrary function f(x) as a continuous sum of exponential functions of the form e^{-itx} ‘ (Springer, 1979 p29).

The Mellin transform and its inverse are (respectively):

$$M_s(\ f(x)\ ) = \int_0^\infty e^{(s-1)\ln x}.\ f(x)\ dx = \int_0^\infty x^{(s-1)}.\ f(x)\ dx$$

$$f(x) = \frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} x^{-s}.\ M_s(f(x))\ ds$$

The Mellin transform is used to determine the distribution of products and quotients of random variables. If X_1, X_2, ...X_n are independent random variables with densities $f_1(x_1)$, $f_2(x_2)$ etc, then the probability density of the product of those random variables is given by the inverse Mellin transform of the product of the Mellin transforms of $f_1(x_1)$, $f_2(x_2)$ etc, namely (Springer. 1979):

$$g(w) = M_s^{-1}\ [\ \Pi_{i=1}^{n}\ M_s(f_i(x_i))\ ]$$
$$= \frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} w^{-s}.\ \Pi_{i=1}^{n}\ M_s(f_i(x_i))ds$$

Even with these transforms there may be no closed form expression or exact series form, in which cases numerical solutions are required. The methods may also be used to determine the probability density function of algebraic functions such as $Y = X_1 + (X_2 + X_3)/X_4$ where each X_i is a probability density function itself (Springer, 1979, p5). The transform methods accommodate only algebraic functions (eg plus, minus, product, quotient and power) and cannot work with trigonometric functions for example (Springer, 1979).

The mathematical approaches to combining random variables are elegant, but excessively unwieldy for all but special cases, and there is little evidence of them being deployed in the design engineering literature. Indeed, even some models which could apply the mathematical approach of random variables (eg Sarper, 1994) do not do so, either because they are too complicated or too sequestrated. It seems unlikely that the mathematical processes will gain use in design engineering given that most designers would find the mathematics excessively tedious and error-prone. Employing a Mathematician alongside the design engineers could be a solution if

industry could come to see the benefits thereof. Alternatively perhaps some of the symbolic computational tools such as *Maple* (incorporated inside *MathCad*[83]) might be able to support the process, failing which numerical solution of the integrals is always a possibility. However once a numerical solution has been used, it becomes more difficult to use that result at another stage of convolution.

4.1.5 **Fuzzy theory**

Another approach is available through *fuzzy set theory* (eg Quelch and Cameron, 1994; Wood et al, 1989 among many others). The fuzzy sets represent partial membership of a set. A typical fuzzy set has the form of a triangular or trapezoidal distribution, described by three or four parameters respectively. The advantage of fuzzy sets is that some relatively simple mathematics can be used to combine sets. Standard *fuzzy arithmetic* operators can be defined, such as add, subtract, multiply, etc. The calculation is based on taking discrete sampling cuts through the fuzzy sets, thereby creating *intervals*, and applying the mathematical operator piecemeal to those intervals to create an output interval. The result of a fuzzy set manipulation is another fuzzy set, not necessarily the shape of either of the inputs.

Although a membership function may be created (fuzzified) from a probability density or histogram, a fuzzy function is not necessarily a probability distribution. The main difference is that the peak of the fuzzy function is given a value of 1, so that the area under the fuzzy function does not sum to unity as a cumulative probability would.

[83] Mathematical software, available from MathSoft (www.mathsoft.com).

4.1.6 **Monte Carlo analysis**

The *Monte Carlo*[84] method is practically the definitive *probabilistic computation* tool (Zhang et al, 1992). In the engineering community it is perhaps best known as a quantitative simulation tool. In the mathematical community is also known for its ability to evaluate integrals numerically (Ross, 1997 p 38). Some applications include Kleijnen (1995)[85] , Zhang et al (1992)[86], Duckworth et al (1998)[87], Basu (1998)[88] and Kostetsky (1994)[89].

Croll (1995) feels that risk analysis methods such as Monte Carlo simulation will spread in usage in engineering projects and that more powerful risk analysis software tools are necessary in order to make the methods "more accessible to the general manager". As Croll points out, one of the major benefits of risk analysis, despite the lack of software tools, is that it "admits to the modelling process ...the existence of uncertainty".

---

[84] *Monte Carlo* randomly samples from input probability distributions, applies the mathematics to compute a single result, and then repeats the process for different random samples thereby building up a histogram for the output.

[85] Kleijnen (1995) motivates for the use in operations research of "sensitivity analysis based on design of experiments and regression analysis, and risk or uncertainty analysis based on Monte Carlo sampling".

[86] Zhang et al (1992) use probabilistic approaches to the risk assessment of mining projects. They report that the most common practices in that industry are sensitivity analysis for deterministic models, and Monte Carlo simulation for probabilistic models.

[87] Duckworth et al (1998) use Monte Carlo simulation to assess financial risk in the water supply industry.

[88] Basu (1998) applied Monte Carlo analysis to schedule (time) risk.

[89] Kostetsky (1994) discusses financial parameters such as cost and time to market, and describes the integration of risk analysis into the development process using Monte Carlo techniques.

4.1.7 **Comment**

The main contenders as probabilistic computation tools would be the algebra of random variables, fuzzy theory, and Monte Carlo analysis. The first is the most precise, but it is too mathematically involved to be practical at early design, and may not even yield a closed form expression. It also only applies to algebraic functions. The fuzzy theory and Monte Carlo methods are more practical tools, and both have gained a following.

Quelch and Cameron (1994) use fuzzy theory to model the fatalities due to gas outflow from a vessel, with randomness in several parameters. They compare the results to Monte Carlo simulation, and it is apparent that the fuzzy set method is an approximation to the results obtained by Monte Carlo. In particularly the fuzzy set method overestimates the weight in the tails.

Similarly Wood et al (1989) compared results from the fuzzy method to those generated by the probability calculus[90] and concluded that there were significant differences between the two methods. They observed that the peak of the distribution created by the probabilistic computation was shifted compared to the fuzzy result. They attributed this to the non-linearity of the mathematical operation, stating that probabilistic computation 'will have similar results to the fuzzy calculus only when the calculation involves linear operations, but not for non-linear operations' (p103). They felt that 'the peak of the [probability] output should be a value that corresponds to the result that would be obtained by using the input peaks as crisp values' (p102).[91] However it is inappropriate to expect that the mode of the probabilistic output should equal the deterministic operator applied to the modes of the inputs, even for linear functions. Instead it would be more realistic to expect the mean of the output to equal the deterministic operator applied to the means of the

[90]The probability calculus method used a numerical algorithm solution to the convolution equations, with triangular input functions.

[91]The α level cuts of fuzzy theory apply interval analysis, and this characteristically ensures the peak of the fuzzy output equates to the operator applied to the peaks of the input membership function. This is because the most-certain fuzzy values (at membership 1) of the inputs are the peaks.

inputs. Note that the mean (weighted average) of a probability distribution is not necessarily the mode (peak of the distribution).[92]

Some reason that since risk analysis is uncertain, therefore there is no need to represent random variables accurately or use sophisticated tools, and consequently that triangular distributions and fuzzy theory are adequate. For example Quelch and Cameron (1994) state that: “It is foolish to believe that we will ever be able to calculate ‘exact’ risk...significant uncertainties are present in any analysis of risk, and a considerable proportion of the analysis is based on subjective expert opinion”. They use this to support their case for fuzzy theory because “the nature of the information required for the analysis corresponds to the limited amount of data often available”, whereas the Monte Carlo method requires that a probability density function be defined for each parameter. However this may be overstating the advantages of fuzzy theory, as both methods require a distribution (the fuzzy membership function[93] or the probability density) and both will accept a triangular distribution. It is undisputed that the information is often sparse, especially at early design stages. Quelch and Cameron believe that triangular fuzzy membership functions are more valid under such conditions than statistical distributions. However they provide no robust justification for this belief. It is difficult to see how a triangular fuzzy membership function should be any superior to established distributions such as the Normal or Weibull that demonstrably represent natural process. In any case, both fuzzy theory and Monte Carlo methods can easily accommodate triangular distributions.

---

[92]The effect of an asymmetrical probability distribution (eg Weibull) is that there is more weight in one of the tails (i.e. upper or lower tail). This moves the mean away from the mode (peak) towards the heavier tail. When such a distribution is convolved with another of any shape, then the additional weight in the tail causes the output distribution to likewise have more weight in one tail. A probabilistic computation or convolution takes into account the weight in the distribution whereas the fuzzy method applies a series of interval analyses which ignore the weight distribution within that interval. Ignoring this information makes the fuzzy method faster, but it also causes loss of information. Hence fuzzy operators will result in broader output membership functions than probabilistic computation on the equivalent inputs.

[93]Although Quelch and Cameron (1994) claim that the fuzzy sets are not probability distributions, it is difficult to see otherwise in the sense that they use them (they appear to be un-normalised (fuzzified) triangular probability distributions).

The main benefit of fuzzy theory is that it is computationally faster than Monte Carlo. However this is at the expense of a partial loss of information about the shape of the output, and special precautions are necessary if the fuzzy set is not monotonic[94] in the interval being considered, whereas Monte Carlo is not constrained in this way. Fuzzy theory can be useful in modelling the extent of uncertainty when the precise shape of the uncertainty is non-critical.[95] Another benefit of fuzzy theory is the provision to process both numerical and textual scales (providing the latter are ordered). While Monte Carlo cannot handle qualitative values, decision tables can.

There is no doubt that the probability calculus and Monte Carlo method are more accurate than fuzzy theory. Nonetheless the probabilistic methods (especially Monte Carlo) are computationally demanding and a few decades ago would have been

---

[94] Standard fuzzy cut sets cannot cope with a membership function that has two peaks, as a horizontal cut then exposes two intervals (four points) rather than a single interval (two points).

[95] Fuzzy afficionados might assert that fuzziness is fundamentally different to probability. In the view of Kantrowitz et al (1997):

> *''Probability statements are about the likelihoods of outcomes: an event either occurs or does not, and you can bet on it. But with fuzziness, one cannot say unequivocally whether an event occured or not, and instead you are trying to model the EXTENT to which an event occured.'*

Such a statement may be overstating the case for fuzzy theory. Many engineering applications of fuzzy theory (such as Wood & Antonsson, 1989) model physical parameters such as length and force with fuzzy sets, but there is no reason why probability distributions should not be used instead, and indeed may be more appropriate. Such physical parameters are not events that either occur or not, but events that have an extent, and they are appropriately modelled by well established statistical probability. Indeed the Normal probability distribution has that name as it has been found to represent variability of many natural biological processes that vary in extent and not as an event. It is therefore difficult to accept the above statement of Kantrowitz et al (1997). It is more likely that the issues are (1) professional rivalry, and (2) easier manipulation techniques for fuzzy- than probability-theory. The rivalry aspect is mentioned by Kantrowitz et al (1997) who state that:

> *'Fuzzy researchers have gone to great pains to distance themselves from probability. But in so doing, many of them have lost track of another point, which is that the converse DOES hold: all probability distributions are fuzzy sets! As fuzzy sets and logic generalize Boolean sets and logic, they also generalize probability. In fact, from a mathematical perspective, fuzzy sets and probability exist as parts of a greater Generalized Information Theory which includes many formalisms for representing uncertainty (including random sets, Demster-Shafer evidence theory, probability intervals, possibility theory, general fuzzy measures, interval analysis, etc.).'*
> (Kantrowitz et al 1997)

These are bold claims and possibly even fanatical, and there is no intent to comment on them here. Others are less enthusiastic about fuzzy logic, for example Brown (2001) notes that in the control application the fuzzy theory is simply a set of logical if-then statements that create an interpolation table, circumventing the need to set up differential equations. Furthermore that such interpolation techniques were available before being incorporated into fuzzy theory. He expresses concern that in safety-critical applications the fuzzy 'arbitrary non-linear truncation and jagged interpolation' makes it difficult to test or assure appropriate behaviour of the control system across its entire range.

beyond reach of many researchers, whereas fuzzy theory was easier to implement on paper without a computer. This pressure no longer exists, as the level of commonly available computer power is adequate to perform demanding computations such as Monte Carlo.

4.2 **DSI probabilistic computation**

This section describes the development of an alternative probabilistic computation method. Probabilistic computation addresses the issue of combining probability distributions with a mathematical operator. The Design for System Integrity (DSI)[96] quantitative method[97] developed here is potentially able to accommodate any probability distribution shape, continuous and discrete, and mathematical expression.

The DSI quantitative method avoids Monte Carlo simulation. Instead it takes the two input probability distributions, converts them to discrete distributions, and then joins them combinatorially using the selected mathematical operator in a process analogous to joint integration, thereby producing the output distribution. There are no constraints on the shape of the input distribution or the operator.

The method is explained with reference to a simple example. Assume that there are two distributions, A and B that are to be added together. A physical interpretation could be that A and B are the lengths of two bars, and we wish to determine the overall dimension when the two bars are laid end-to-end. This is the typical tolerance stack-up problem faced in machine design and manufacturing. The identical mathematical problem occurs in financial risk modelling (cost summation), and risk assessment of project management schedules (time summation).

---

[96]The author terms this the Design for System Integrity (DSI) method as the intended application is to assess the integrity of a designed system, at greater detail than is provided by single-point estimates of failure probability.

[97]There is a separate DSI qualitative method which is described in a subsequent chapter.

Each of dimension A and B has some variability and this is represented by a discrete probability distribution. These data might typically be obtained by measuring lengths of several samples and producing a histogram. The dimensions A and B are shown in Figures 4.1 and 4.2. The given probabilities are discrete probability densities, the sum of which totals 1.00 in each case.

Distribution A	
Dimension [mm]	Probability
10	0.2
20	0.4
30	0.3
40	*0.1*

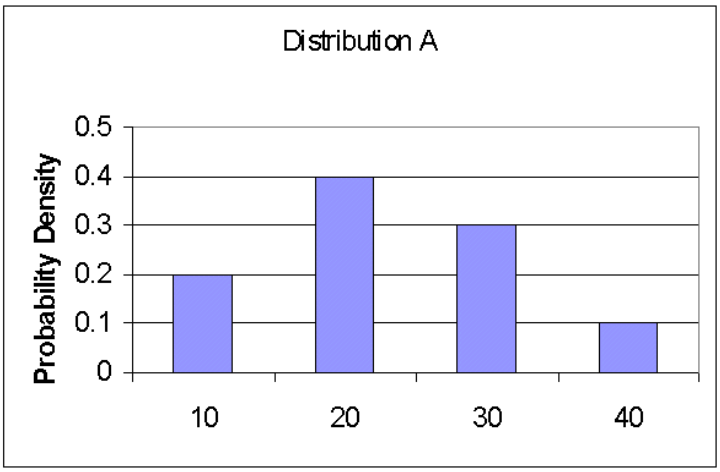


Figure 4.1 Input distribution A.

Distribution B	
Dimension [mm]	Probability
10	0.6
20	0.35
30	0.05

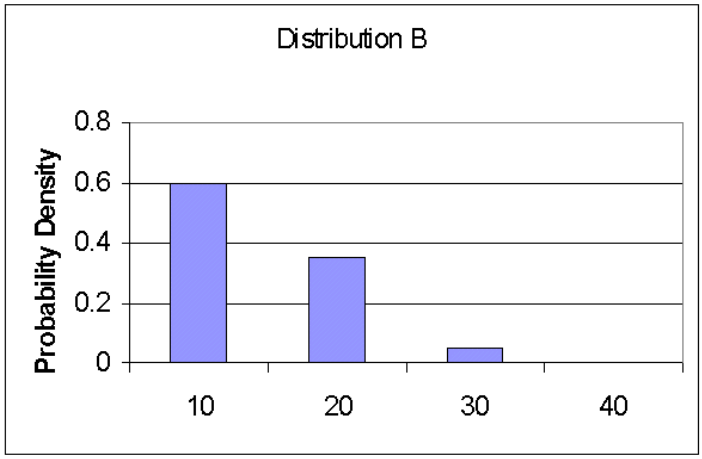


Figure 4.2 Input distribution B.

If this addition were done in a deterministic manner, then using the mode, the result would be:

Dim C = Dim A + Dim B

= 20 + 10

= 30 [mm]

In the case of probabilistic computation it is necessary instead to determine each of a number of individual outcomes. For each input (say Dim A = 10), it is necessary to consider every output of Dim B (10, 20, 30) and work out the resulting value of Dim C, as well as the probability of that event. As the relationship is additive in this case, the values of Dim A and Dim B are added. The principle is easily extended to other relationships, such as subtraction, multiplication and division among others. However the joint probability for the event is always the product of the two constituent probabilities, assuming the two inputs are independent[98]. The full table of outcomes is shown in Table 4.1

Dim A		Dim B		Dim C		Prob A		Prob B		Joint Prob
10	+	10	=	20		0.2	x	0.6	=	0.12
10	+	20	=	30		0.2	x	0.35	=	0.07
10	+	30	=	**40**		0.2	x	0.05	=	0.01
20	+	10	=	30		0.4	x	0.6	=	0.24
20	+	20	=	**40**		0.4	x	0.35	=	0.14
20	+	30	=	50		0.4	x	0.05	=	0.02
30	+	10	=	**40**		0.3	x	0.6	=	0.18
30	+	20	=	50		0.3	x	0.35	=	0.105
30	+	30	=	60		0.3	x	0.05	=	0.015
40	+	10	=	50		0.1	x	0.6	=	0.06
40	+	20	=	60		0.1	x	0.35	=	0.035
40	+	30	=	70		0.1	x	0.05	=	0.005

Table 4.1: Outcomes for all combinations of inputs A and B.

[98]The DSI method developed here assumes independence of the input random variables. This is frequently the case in many applications. Where it is not valid, the DSI qualitative map may be used to correlate the two variables.

For every outcome of Dim C there is an associated probability (Joint Probability). Furthermore, it may be observed that certain outcomes of Dim C occur more than once (eg 40 occurs three times). The probabilities for common outcomes must then be summed. This results in a condensed table that may be charted as in Figure 4.3. This gives the probability density for the outcome. In our example this corresponds to the total length of the two parts.

Dim C [mm]	Prob density C
20	0.12
30	0.31
40	0.33
50	0.185
60	0.05
70	0.005

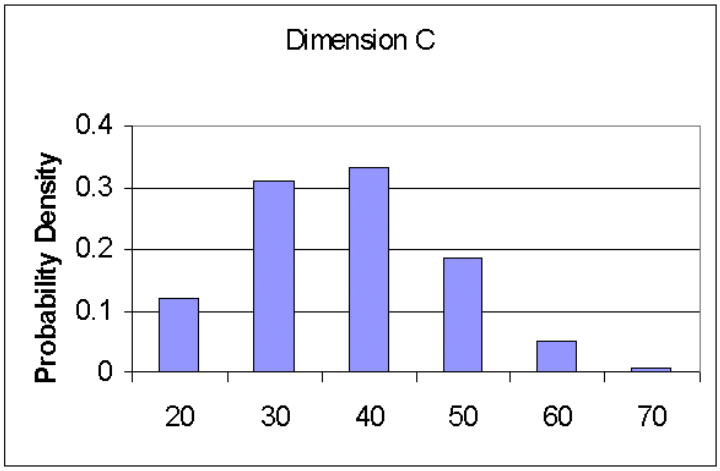


Figure 4.3 Output distribution C.

These results show that the most likely outcome is 40 [mm], with a probability density of 0.33. More importantly from a risk assessment perspective, the result shows what other values of output may be expected, and the probability of each. It becomes possible to determine the probability of extreme events.[99] Deterministic methods are not able to provide this level of detail.

In terms of managing risk and integrity, the chart that is of greatest interest is the cumulative probability. This is simply determined as the running sum of the probability density values already calculated. It is shown in Figure 4.4.

[99]For example there is a 0.005 probability (0.5%) of an outcome value of 70 [mm], and 0.12 probability (12%) of the minimum value of 20 [mm].

Dim C	Prob density C	Cum prob C
20	0.12	0.12
30	0.31	0.43
40	0.33	0.76
50	0.185	0.945
60	0.05	0.995
70	0.005	1

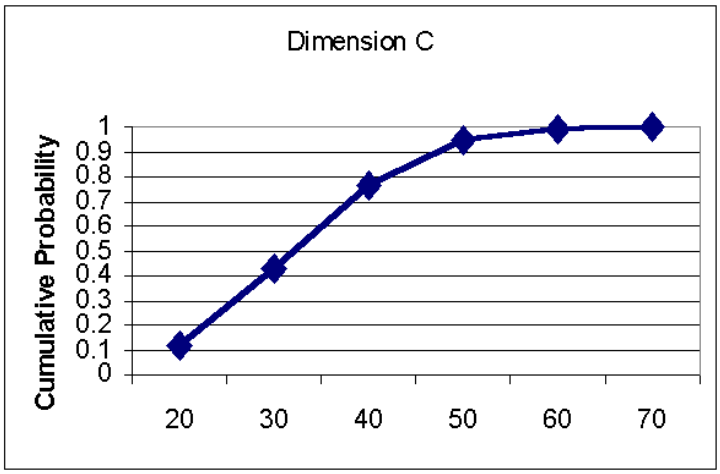


Figure 4.4: Cumulative probability for output.

The cumulative probability shows the probability of getting an outcome from zero UP TO AND INCLUDING the point of interest.[100] Other percentiles (eg 10% and 90%) may be found in a like manner.

4.3 **Implementation**

DSI imposes no constraints on the shape of the input distributions. As the number of intervals is increased, so it becomes possible to approximate continuous distributions. The ability to handle other discrete distributions is retained, so the method is able to perform mathematical operations on any two input distributions regardless of their type.

The discrete approximation may seem a limitation, but this needs to be seen in the context that even the Monte Carlo method makes discrete approximations in that it takes random values and sorts outputs into histogram bins. The major advantage of

[100]For example, there is a 0.43 probability of the outcome being 30 [mm] OR LESS. The median (0.50 probability) can also be determined from the cumulative chart. In this example it is about 32 [mm]. In other words, 50% of parts will be 32 [mm] or less, and 50% will be longer.

the DSI method over Monte Carlo is that there is not the uncertainty as to whether sufficient samples have been taken to give an accurate outcome. For more accuracy, the DSI method simply requires finer intervals. However the shape is always correct, to within the resolution provided by the interval width, whereas Monte Carlo cannot reliably identify the shape until significant iterations have been completed.

Finer intervals naturally require greater computational effort. If two input distributions are each approximated by 100 intervals, then the number of outcomes to determine is the product of the input interval counts, i.e. 100x100 = 10 000. However this type of manipulation is well within the capabilities of current computing technology, so this presents no particular barrier. DSI permits coarse intervals to be used at first, and refined later when the computational cost is warranted. Even with coarse intervals the output shape is evident, whereas this benefit is not likewise obtained with low iterations of Monte Carlo.

The DSI method is readily extended to accommodate various mathematical relationships between the input distributions. Currently the software is configured to operate on up to three input parameters across a range of arithmetic and logical operators. Note that many relationships can be computed in pieces, eg A+B+C+D = (A+B)+(C+D) as would be done in the deterministic approach using a hand calculator. In principle DSI could be implemented with more than three inputs.

4.3.1 **Software implementation**

Implementation of the DSI algorithm required some slight adaptions to the simplified method described above. An underlying assumption in the method is that the individual joint outcomes have a piecewise uniform distribution. In other words, a joint probability is evenly spread across the (small) outcome interval. It is therefore necessary in the general case to determine the small outcome interval itself, and not just the centre point. The interval over which to distribute the fragment of joint

probability is determined by applying the mathematical operator to both the upper and lower bounds of the input intervals. Effectively this means that the product operator results in larger interval widths in the output array than does say addition. The necessity to determine the limits to the interval adds slightly to the computational effort compared to the simplified representation given above.

A further complication concerns output interval sizes. The simplified example used input and output bins that were all multiples of ten. However in the general case the input bins may be of different sizes. Even the output bins themselves will not always be constant (applies to product and division especially). Then it becomes necessary to proportionally sort the output array (at a computational cost) assuming that the probability is distributed uniformly across the small intervals. An alternative sorting method is also provided in the software, whereby the output is a point with lumped probability. This is a simpler and faster algorithm than proportional sorting, though it gives results which are very similar to that of the proportional sort.[101]

A significant part of any software development is the user interface, and no less in this project. The algorithm has been implemented in the Delphi programming language, for the built-in user-interface tools that it provides.[102] The resulting software implementation of the algorithm is a stand-alone executable that runs under the Windows operating system. The software is described in further detail in a subsequent chapter.

---

[101]The lumped point approach can result in co-incidental interference on the output histogram, i.e. one histogram bin stealing weight from a neighbouring bin because of the set-up of the bins.

[102]Delphi is an object oriented programming (OOP) language. Unlike an expert system it is a procedural language in that the order of the statements IS important. The software is noted for the good support it provides for the creation of user interfaces. It has similar features and capability to C++, which would be its closest competitor in terms of functionality. Delphi provides blank responses to user actions such as mouse clicks. However it requires significant programming to generate outputs, and is not so much a shell that the developer can place data into, as a blank canvas on which everything must be created. This gives flexibility to the type of applications that can be developed with Delphi, at the expense of requiring the developer to code that functionality. Delphi compiles the user's code into an executable file that runs on Windows operating systems. Being fully compiled the executable is very much faster than applications like Visual Basic. Once compiled the executable does not require the continued presence of the Delphi software, i.e. it is fully independent, unlike some other programmes. Also, unlike several other development packages, Delphi does not require a license for each copy of an application that is deployed. Delphi is available from Borland - Inprise Corp, 100 Enterprise Way, Scotts Valley CA 95066-3249 USA.

4.3.2 User involvement

In using the software the user first specifies the two input distributions. Several types of continuous distributions are supported, including the Uniform, Triangular, Weibull, Normal, and Beta (and therefore also the uniform and exponential). For these the user has to provide only the necessary parameters to describe the distribution. In addition the user controls the number of discrete intervals and the starting point.[103] The software then calculates the parameter range (eg time intervals) and the probability of each interval. Alternatively the user can select to load a probability distribution from a text file, which could represent recorded data or the result of a prior processing stage. The user can also manually enter the probability histogram. Chapter 9 and Appendix 1 describe the user interface in more detail.

Confidence levels

Another method of describing the input distribution is provided in the form of confidence levels. Under this method the user provides two or more estimates of confidence. For example, the user might believe that there is a 5% chance that the result will be less that or equal to 800 [hrs or $ or other relevant units], and a 80% chance of a result less that or equal to 1000 [units]. These represent confidence limits. The user may enter as many such estimates as required, two being the minimum. From these values the software determines an appropriate Normal or Weibull distribution, using a least squares fit.

The advantage of using confidence levels is that it forces the user to think in a probabilistic rather than deterministic fashion. If the input variable is well defined or

[103] *Loaded tails* - If the user's selection of range has truncated the distribution significantly, then it is likely that the first and last intervals will show higher than expected probability. This is because the software is configured to load the lower and upper tails onto the first and last intervals respectively. These loadings should not be edited out by the user, as they reflect reality. They are however a conservative estimate of the tail, in that the entire tail (eg upper tail up to positive infinity) is loaded onto one finite interval. This creates something of a modelling artefact when the joint probability is determined. The solution is to redefine the intervals, selecting more intervals and wider, so that the tail residuals are less prominent. Of course with bounded distributions such as the Beta and the Triangular there are no infinite tails so the problem should not need to occur unless the user specifically wants to truncate such a distribution.

some prior knowledge exists regarding it, then the confidence limits will be close together and produce a distribution with low spread. However if the user has large uncertainties of belief, or lack of knowledge or data on which to base the estimate, then the confidence limits will be wider spaced and the resulting distribution will have greater spread.[104] This level of detail is simply not available in conventional deterministic analyses that only use the mean value and ignore the uncertainties. Although hard data may be sparse at early design, there may be robust beliefs in the minds of the development team, possibly based on previous experience. Confidence levels provide a means to incorporate these beliefs into the probabilistic computation methodology.

Once the input distribution has been generated, it is charted and shown in a box where the user can edit it if necessary. A normalisation function is provided, and intended for where the user has entered a histogram manually using counted incidences. Likewise a second input distribution is created, which may be different to the first. Once the two input distributions are defined, the user selects the mathematical operator. The system then calculates the joint probabilities and then sorts them into the output bins. Finally the system displays the output density histogram, as well as the cumulative probability. The results may be saved and used as input to subsequent modelling sessions. A simplified flowchart of the software processes is shown in Figure 4.5.

[104]When using a confidence level estimates, it is necessary to apply some discernment as to the selected interval size. Fine discrete intervals will not necessarily have an accuracy advantage over coarser intervals. Finer intervals may be more visually appealing, but they will only provide additional accuracy if there are valid reasons for supposing that the underlying processes are following a particular distribution. Having said this, there are indeed distributions such as the Normal and Weibull which do provide the necessary underlying justification in certain cases.

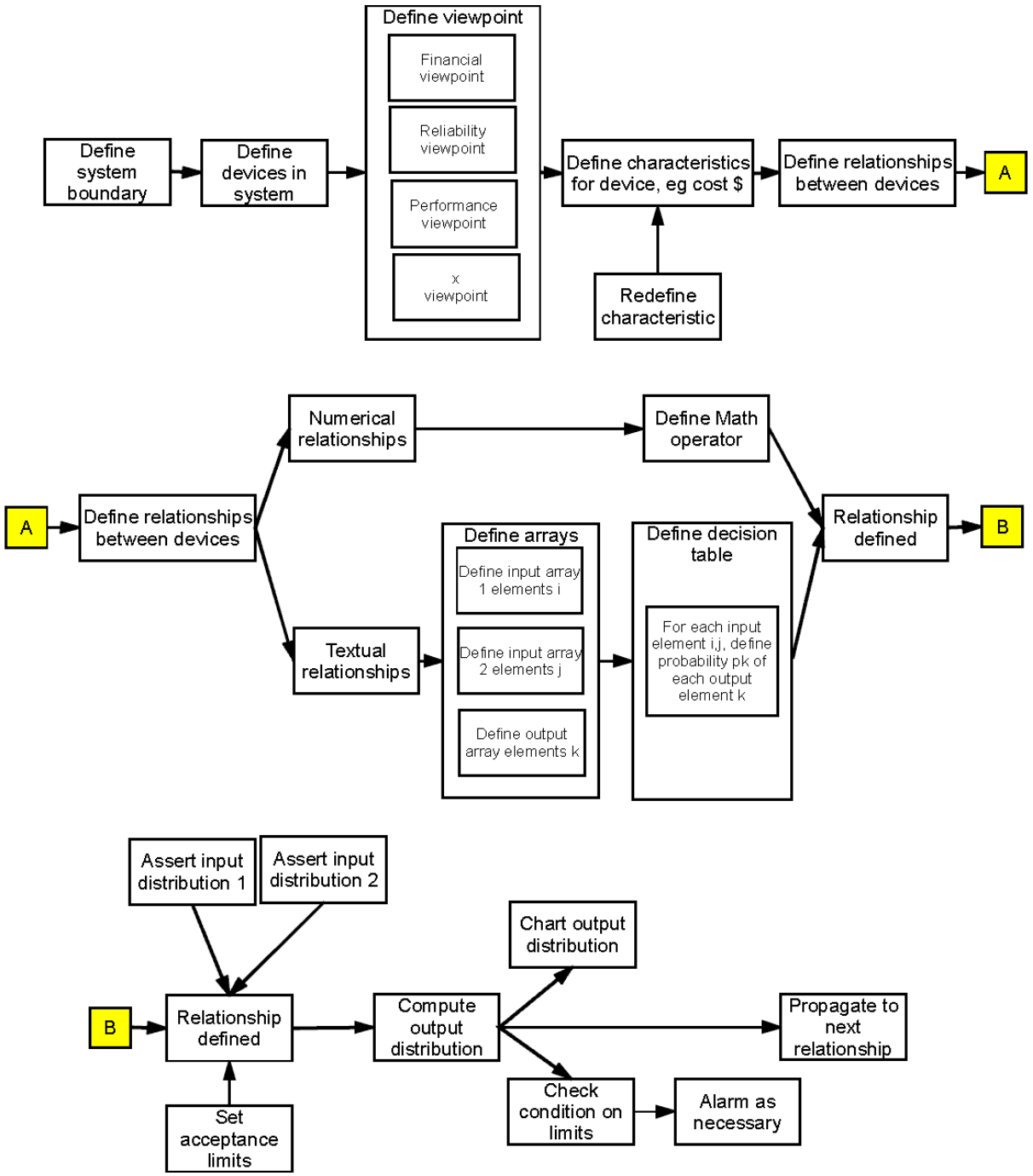


Figure 4.5: Flowchart for the DSI method.

4.3.3 Example results

In principle any mathematical operator with two input parameters can be modelled. Operators such as addition, subtraction, product, division, greater and lesser (among others) have been provided. Minor additions to the source code could create other operators as necessary.

Sample applications of these operators are shown below. In each case the input distributions are Weibull (eta=150, eta=1.6) and Normal (mean=200, stddev=45). The figures show the probability density for both input distributions and the output. The output is the darker line with markers.

Sum

The sum of two distributions is a common operator in many situations. It is typically required for applications such as tolerance stack up, financial risk, and time (project management schedules).

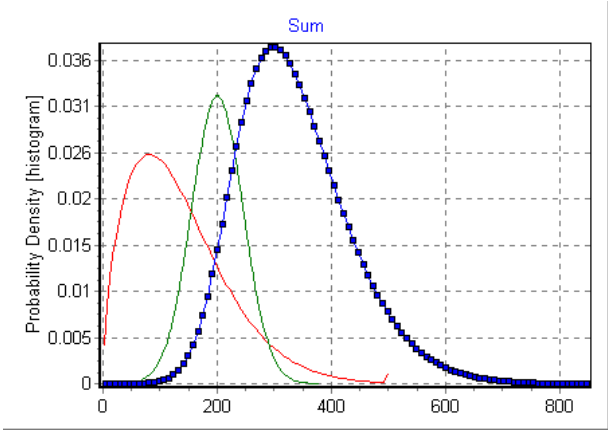


Figure 4.6: Summation of two input probability densities.

Subtraction

The difference between two distributions is required in similar cases to addition. An example is investigating the fit of two components, eg a shaft into a hole.

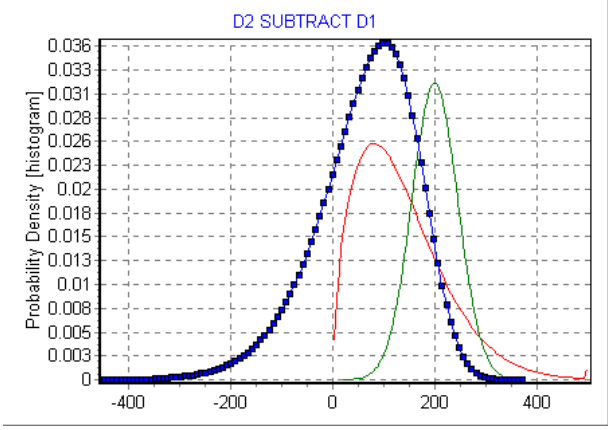


Figure 4.7: Subtraction of two input probability densities.

Product

The product operator provides the probability of various outcomes from distributions that are multiplied together. Applications include modelling mathematical relationships such as power = torque x speed, as well as financial risk analysis.

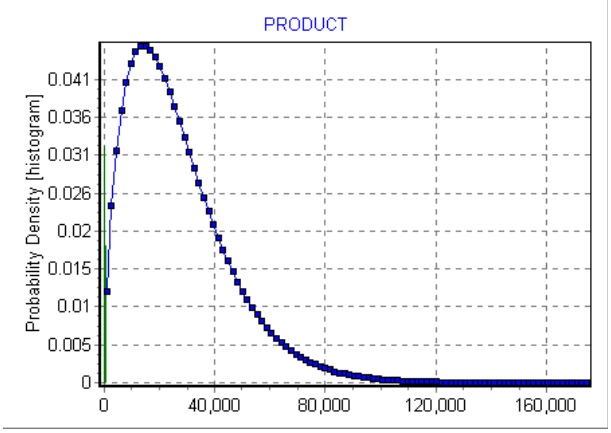


Figure 4.8: Multiplication of two input probability densities.

Division

The division operator is used in similar applications to 'product' operator.

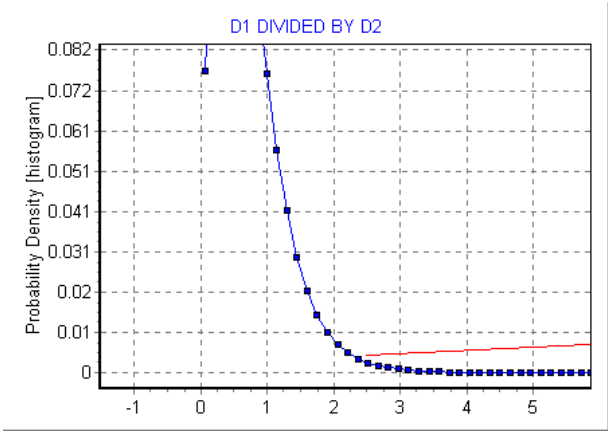


Figure 4.9: Division of two input probability densities.

Lesser

The 'least' or 'or' operator is used to model failure of a machine from multiple failure modes. The machine fails when either of the failure modes occurs. This is not the same as the 'subtract' operator.

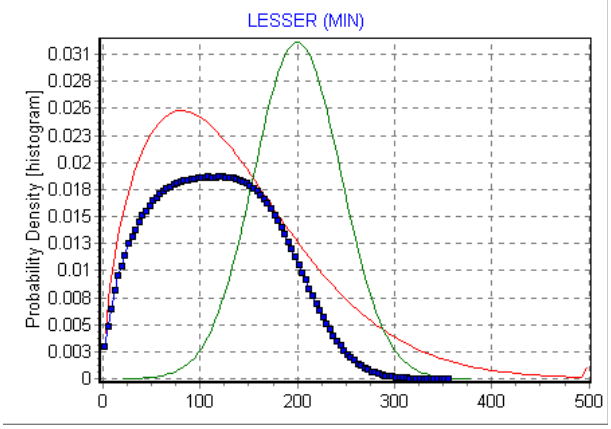


Figure 4.10: Lesser of two input probability densities.

Greater

The ‘greater’ or ‘and’ operator provides an outcome which is the probability of the last item to fail. It is not the same as the ‘sum’ operator.

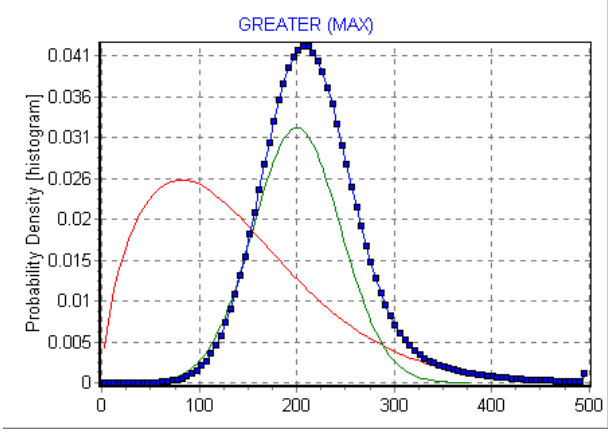


Figure 4.11: Greater of two input probability densities.

The above illustrations show the type of result possible with the methodology. Some of the output distributions bear a resemblance to the Normal distribution but the resemblance is superficial.

4.3.4 Interpreting results

The above illustrations give the *probability density* as a histogram for the two input functions and the output function. While some distributions may appear to have greater area under the curve, this is an artefact as the bin widths of the histograms are not generally the same, and this affects their relative heights. DSI provides the option to display the data using true density if that is preferred.

The probability density charts are useful in determining the shape of the resulting distribution, and the *mode* (the peak of the distribution, which is the most commonly occurring value). The charts also provide descriptive insight into the behaviour of the tails in the output distribution, and are intuitively easier to interpret than the cumulative charts.

In terms of assessing integrity and risk, the cumulative charts are usually more valuable. In particular, they permit the area in the tails to be determined. The two types of probability distribution are shown in Figure 4.12 and 4.13, density and cumulative, for the same output distribution ('lesser' outcome in this case).

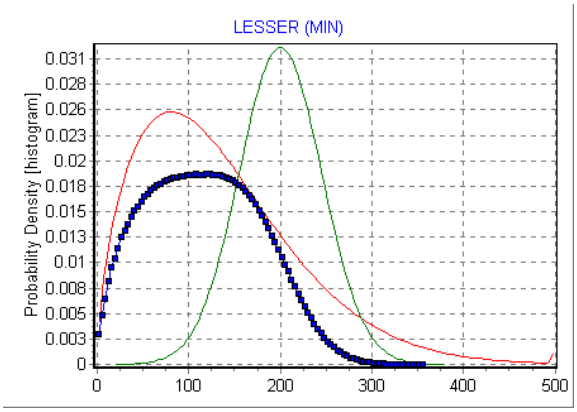


Figure 4.12: Probability densities for 'lesser' operator

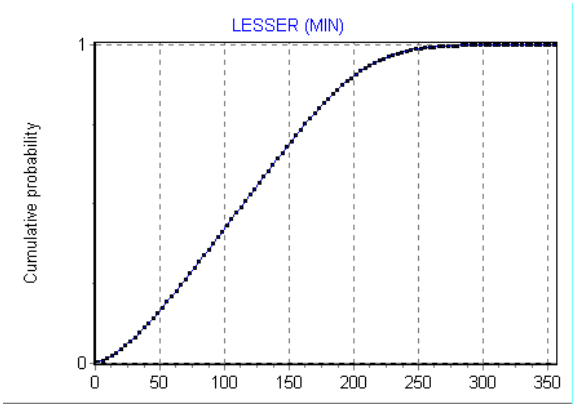


Figure 4.13: Cumulative probability for 'lesser' outcome.

The *cumulative distribution* is characterised by starting at zero and ending at one on the probability scale. It also often has a sigmoid shape, though this shape is apparent for all the distributions that have a bell-shaped density. The sigmoid shape is necessary but insufficient evidence for the Normal distribution. It is difficult to infer the shape of the density distribution from the cumulative distribution, at least by inspection, and the cumulative should not be used for this purpose. Instead its value is in quantifying the outcome in percentiles, of which the *median* is the most important. [105] The tails are especially important in risk assessment since they quantify the severity of the outside risks.[106]

---

[105]The median is the value [eg time] at which 50% of the outcomes have occurred. In the example above the median is at 114 [hr]. This can only be determined from the cumulative distribution, as it is not available by inspection of the density. The density shows the mode, and this does not necessarily correspond to the median. The other commonly used statistic is the *mean*, which is the weighted average (weighted by probability) and this is different again to the median and the mode. For symmetrical distributions like the normal, the mean, median and mode are coincident. However this is not the case with skew distributions. The preferred statistics are the mean and median.

[106]For example, the data could correspond to the reliable life of a machine that consisted of two subsystems, and the whole was deemed to have failed when one part failed. The input distributions correspond to the failure probability of the constituent parts, and the output to the expected life of the assembly. If the required life of the assembly is 50 [hr], then the cumulative chart shows that the probability of a failure in the time up to and including 50 [hr] is 0.167. In other words

4.4 Validation against algebra of random variables

In this section the DSI method is validated against the algebra of random variables, as the latter is the definitive mathematical approach. The first part provides the validation, and the second explores the sensitivity of the DSI method to low sample sizes. The validation of the software is described in chapter 9, along with a discussion of the terms validation, verification and testing.

4.4.1 Mathematical validation of DSI algorithms

The DSI software provides a new approach to quantitative probabilistic computation, and there is a need to ensure that the algorithms used to achieve this are mathematically valid. It would be beyond the scope of this project to attempt a formal proof. However the algorithms can be demonstrated to be valid, in the sense of being defendable, and this is discussed next. The approach used was functional testing (Powell, 1982) which is 'an application of test data derived from the functional requirements without regard to the final program structure' (p 30).

Functional testing was performed using a test sample. The test consists of the quotient of two identical uniform (0,1) distributions. This problem is solved by Springer (1979) (p 95) using the Mellin transformation and the *algebra of random variables*, and the resultant density distribution is:

h(x) = 0.5 for 0<= x<= 1 and
= 1/(2x^2) for x>= 1

This is a mathematically explicit equation for the distribution. The distribution has a most unusual and somewhat unexpected shape, being flat with a sudden tail, see

16.7% of the machines would be expected to fail. This information would be valuable to a manufacturer in determining the exposure to warranty claims. The upper tail can be determined in a slightly modified manner. For example, the probability of a machine life exceeding say 250 hrs is determined by noting that the probability of failure before that time is 0.987, and therefore the probability of a longer life is one less this, namely 0.013 (1.3%).

Figure 4.14. This test was selected because of the ready visual identification of this unusual shape. By comparison summation operations tend to produce a bell shaped distribution which often bears superficial resemblance to the Normal distribution, so distinguishing on visual grounds alone is often difficult.

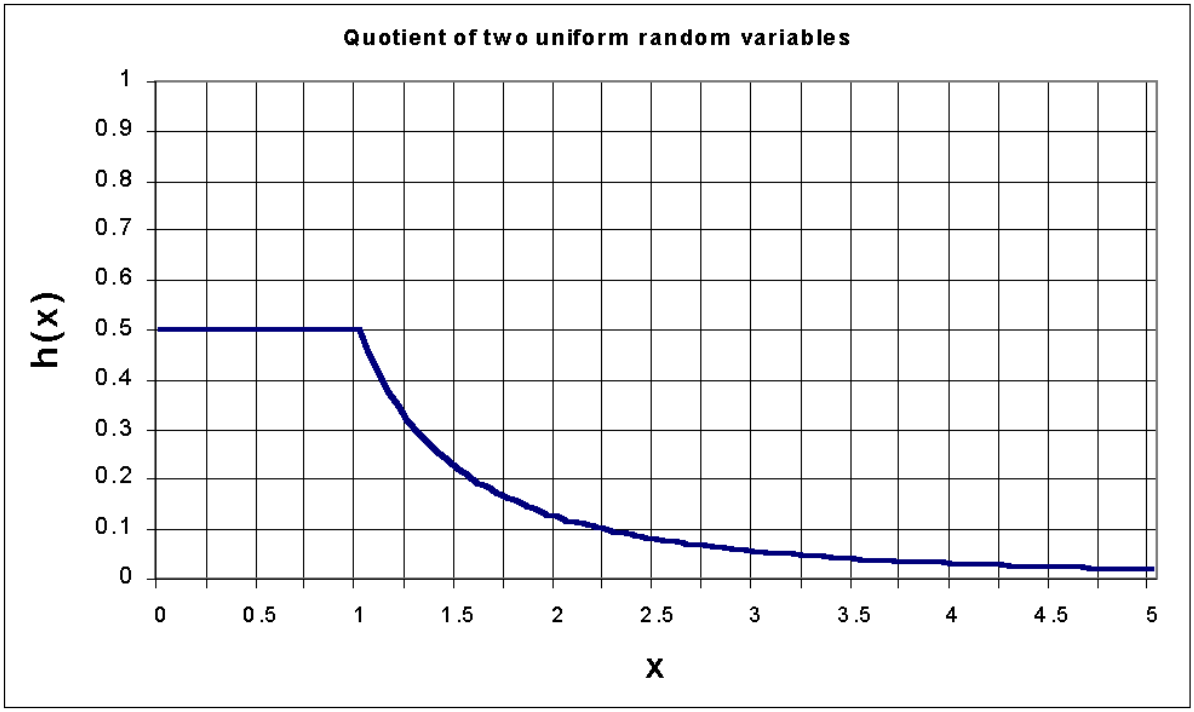


Figure 4.14. Probability distribution resulting from the quotient of two identical uniform distributions (0,1), as determined by the algebra of random variables.

The DSI results for the test are shown in Figure 4.15. This chart shows true density whereas the default display in DSI is histogram. The setting may be changed in the software by the user. The large peak at far right is caused by all the residual upper tail being lumped onto the last interval. It is apparent from inspection that the DSI algorithm produces results with the same shape as required by the algebra of random variables: the plateau is in the right place, and the tail is subjectively correct. Note that DSI lumps all the residual upper tail onto the last interval, hence the spike at the end of its distribution. That spike is therefore permissible. It is futile superimposing the two charts as the difference is at most at the sixth decimal point. A chi square goodness of fit test was performed (see Appendix A2.1). The significance

of fit was at least 99.99999% (the computational limit of the chi square function in MicroSoft Excel). This is an exceptionally good result and indicates a very close match between DSI and the algebra of random variables.

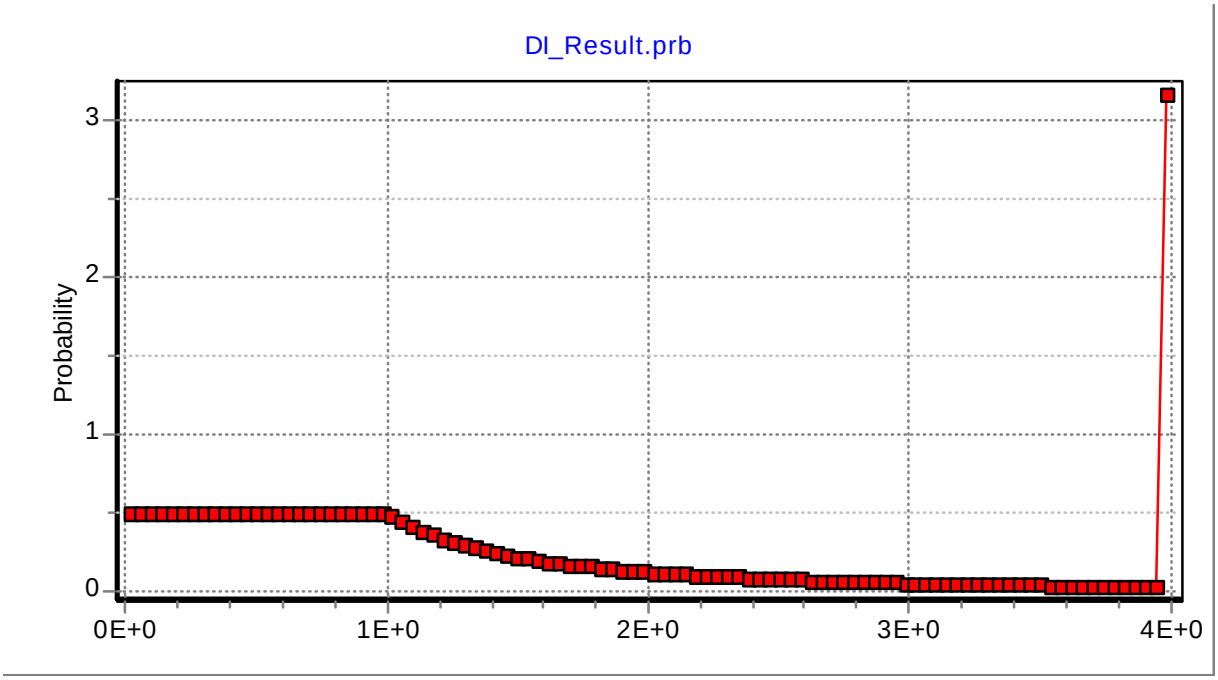


Figure 4.15. Probability distribution resulting from the quotient of two identical uniform distributions (0,1), as determined by DSI.

It is concluded that the DSI method is faithful to the algebra of random variables beyond reasonable doubt, at least for the test sample and the quotient operator.

In principle additional test cases could be devised for this and the other mathematical operators included in DSI. However the above test has already confirmed that the DSI method is in certain cases a valid probabilistic computation method, and it is submitted that this is sufficient for current purposes. The scope of this thesis was that such a method was possible.

4.4.2 Robustness of DSI to low resolution simulation

DSI has been shown to deviate to an insignificant degree from the algebra of random variables, when using about a hundred points. How robust are its algorithms when fewer data points are used?

Its behaviour was explored using the following test. The test is again the quotient of two uniform (0,1) random variables. The algebra of random variables is the reference method and it predicts the results shown previously in Figure 4.14. A DSI model with 100 data points has been demonstrated above, and it is shown that is produces an excellent match to the algebraic method. Here the test is a DSI model with only 10 data points. This is a sizable reduction in modelling resolution and a large degradation might be anticipated in the accuracy of result. The DSI result (points) is shown in Figure 4.16, superimposed on the Mellin result (smooth curve) from the algebra of random variables.

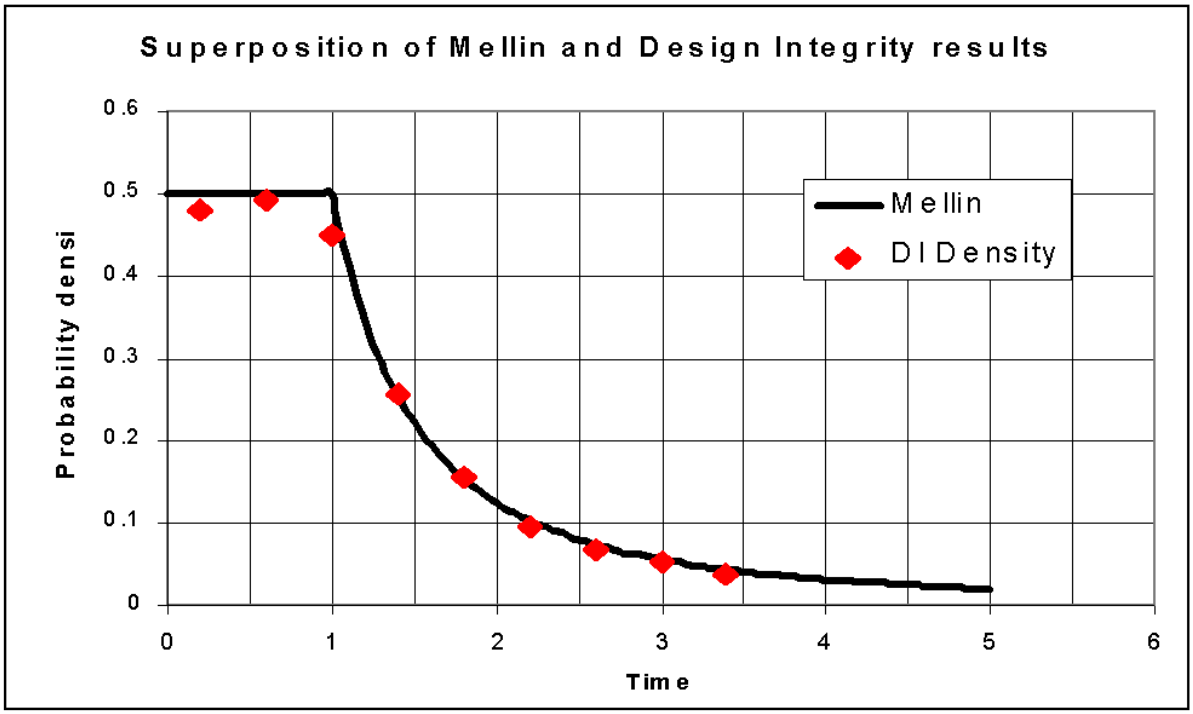


Figure 4.16. Probability distribution resulting from the quotient of two identical uniform distributions (0,1), as determined by DSI, with only 10 data points.

Visual inspection of the figure suggests that even at this very low modelling resolution the DSI method produces results that are close to the Mellin result. A chi square test was used to measure the goodness of fit (see Appendix A2.2). The test shows that there is no significant difference between the DSI and Mellin results at 99.99999% significance. This result is statistically highly significant and indicates that the DSI method provides a robust probabilistic computation even when modelling resolution is low.

4.5 **Comparison with Monte Carlo analysis**

A Monte Carlo method was written in the software to provide a benchmark. The method uses the same input distributions (frequency histograms) as the DSI method. It converts these into a probability density weighted scale, from which a random selection is made.[107] Values with greater probability density are more likely to be selected, so the tails are not over-represented. In this regard the method operates similarly to the Latin Hypercube sampling algorithm.

4.5.1 **Monte Carlo benchmark test**

The benchmark test was done on the subtraction of two Weibull distributions. The details are shown in Table 4.2, and the results in Figures 4.17 to 4.20.

[107] The random values from the input distributions are combined according to the mathematical operator selected, and the result placed in an output bin. The process is repeated many times to build up the output distribution.

Input Distribution 1 (D1):	Weibull, characteristic life 2500, shape factor 2.5, number of intervals 50, starting from 0.
Input Distribution 2 (D2):	Weibull, characteristic life 6400, shape factor 4.0, number of intervals 50, starting from 0.
Operator:	Subtract, D2-D1
Output Distribution (X):	Number of intervals 90, starting from -3000.

Table 4.2: Benchmark test using subtraction of two Weibull distributions.

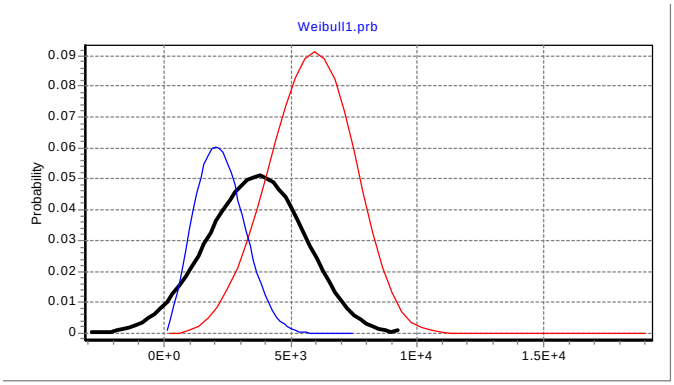


Figure 4.17: Results from DSI numerical benchmark with smoothing (proportional sort). Computation time was 4.340 s.

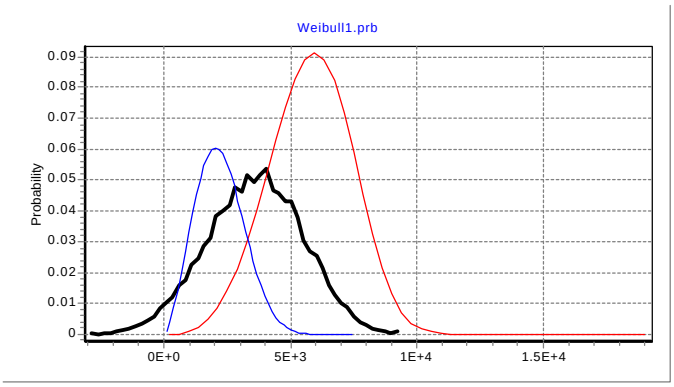


Figure 4.18: Results from DSI numerical benchmark, with no smoothing. Computation time was 1.600 s.

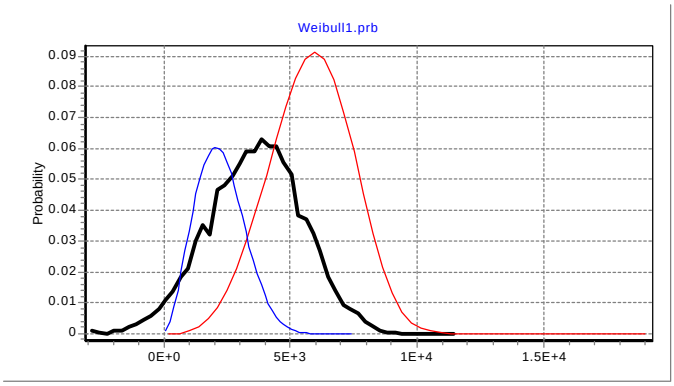


Figure 4.19: Results from Monte Carlo algorithm using 10000 output bins. Computation time 5.220 s.

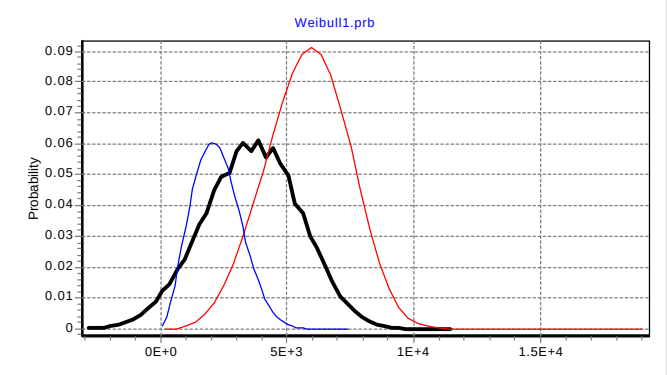


Figure 4.20: Results from Monte Carlo algorithm using 1 000 000 output bins. Computation time 2 min 33.070 s.

4.5.2 **Discussion of benchmark results**

Comparing Figures 4.18 and 4.20 for DSI and Monte Carlo respectively, it is evident that both methods exhibit similar interference artefacts. These are caused by some outcome bins being more likely to receive hits in turn due to the regular spacing of the discrete inputs. DSI and Monte Carlo produce similar results, though DSI is computationally quicker. The optional smoothing method in DSI operates for Figure 4.17 and removes these artefacts, though there is a slight computation cost.

Monte Carlo analysis requires large runs (Figure 4.20), of the order of a million output results, in order to obtain charts with similar accuracy to DSI The DSI method gives comparable results in a significantly shorter time.

4.5.3 **Comparison with 'Analytica'**

'Analytica' is a commercially available system that provides Monte Carlo simulation. Analytica uses an *influence diagram* as the graphical user interface. The user places the blocks and connects them with arrows, then defines the mathematics of the links.

Figure 4.21 shows a composite screen of various windows. The 'Bulb Characteristics' box provides for the ability to enter different bulb life parameters and perform a type of 'what-if' analysis in a batch file process. DSI does not have this capability, though it is relatively easy in both systems to adjust the bulb parameters manually to achieve the same effect. This model is a modified version of 'Failure Analysis.ANA' supplied with Analytica.

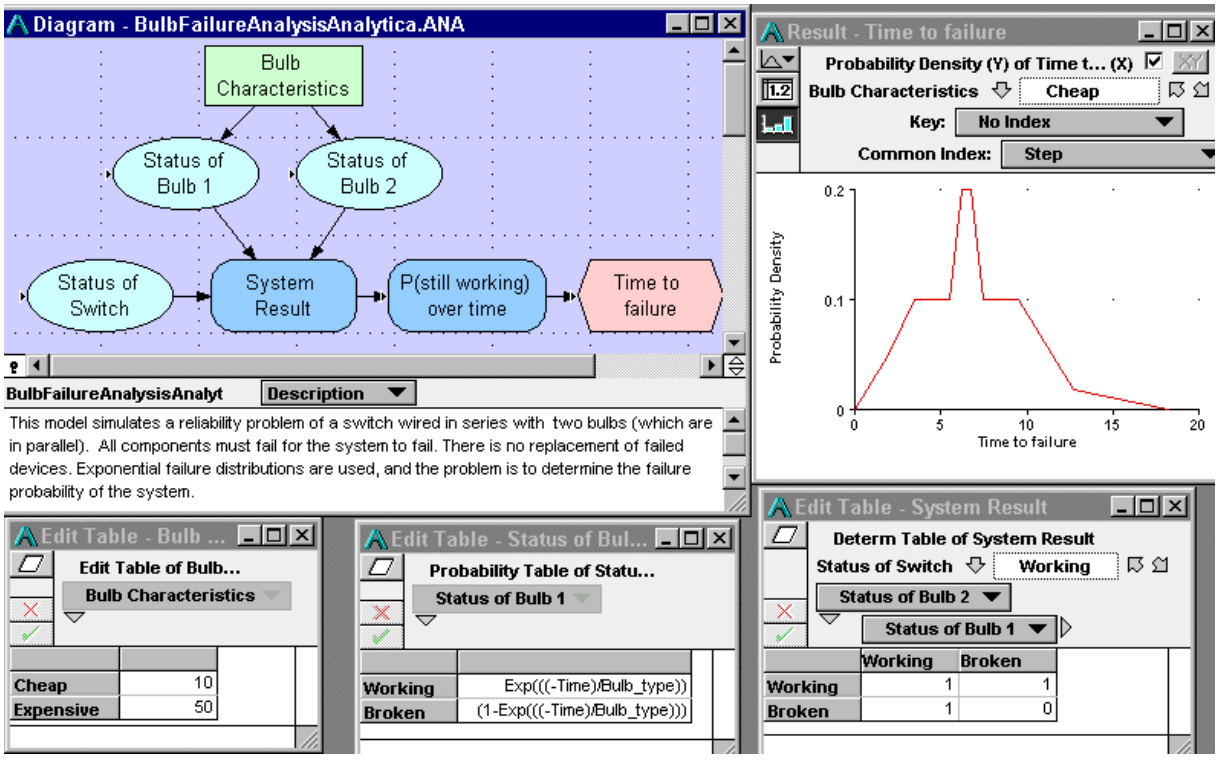


Figure 4.21: Reliability model as it appears in Analytica.

The influence diagram itself is at top left (titled 'Diagram - BulbFailureAnalysisAnalytica.ANA'). At middle bottom ('Edit Table - Status of Bul...') are the exponential distributions for the bulb failure probability. Similar windows (not shown) define the distributions for the other bulb and the switch. The state of the system is defined in the window at bottom right ('Edit Table - System Result') as a decision table. That window shows whether the system is working (1) or broken (0) given the state of Bulb 1, Bulb 2 and the Switch (each of which may be working or broken). The illustrated window shows only a slice through the decision table for switch = working, and the next slice would be a mouse click away. The table is called a 'Deterministic Table' by Analytica. It is important to note that there are not probabilities associated with this table (in computational terms it is a compact form of nested 'if' statements).

A simple reliability model is used to illustrate the resolution differences between DSI and Analytica. This model simulates a reliability problem of a switch (F(t) = exp(-t/1000)) wired in series with two bulbs (F(t) = exp(-t/10)) which are in parallel. All components must fail for the system to fail, and there is no replacement of failed devices. Exponential failure distributions are used, and the problem is to determine the failure probability of the system. This model is a modified version of '*Failure Analysis.ANA*' supplied with Analytica. The DSI representation is shown in Figure 4.22 and the Analytica version in Figure 4.23.

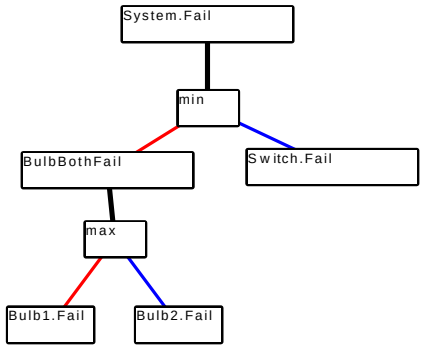


Figure 4.22: DSI view of reliability of two bulbs (in electrical parallel), in series with a switch.

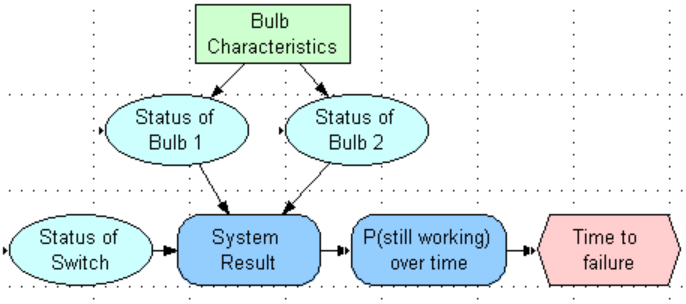


Figure 4.23: Reliability model as it appears in Analytica.

The results for the simulations are shown in Figure 4.24 for DSI and Figure 4.25 for Analytica. Computational times were similar. It is evident that Analytica gives the coarser resolution. It also does not have the extended upper tail, corresponding to the less likely failure of the switch, for reasons which can only be surmised.

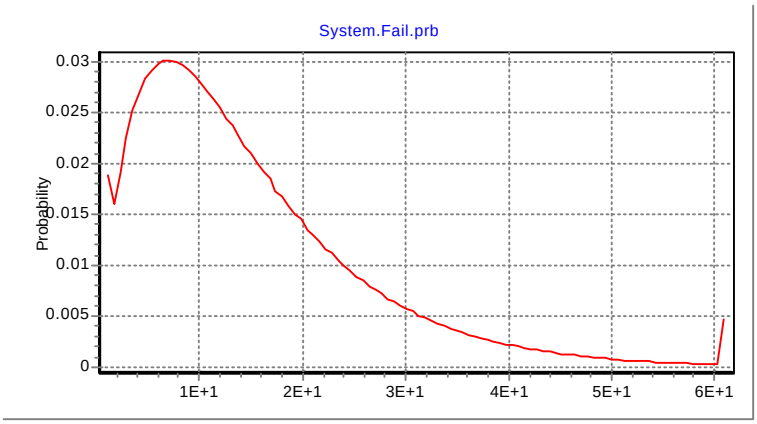


Figure 4.24: System reliability by DSI, mean 15 hrs, and median 12 hrs.

It is therefore evident that DSI produces higher resolution results, and more faithful shape of distribution than Analytica. The computational time is similar. However DSI goes further in explicitly providing support for multiple views in functional modelling.

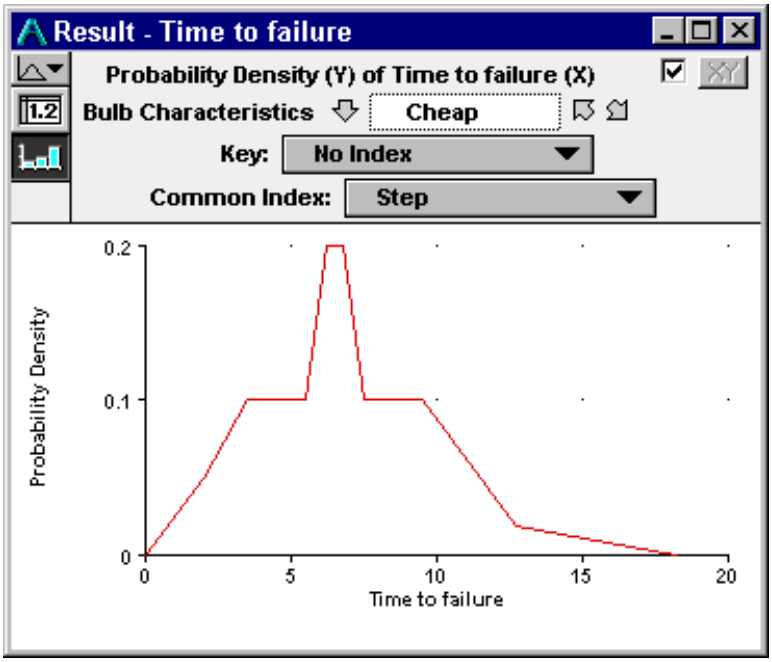


Figure 4.25: Results from Analytica simulation for system life.

Artefacts

Any Monte Carlo method, including that used in Analytica, creates more noise in the results than the DSI method. For example, a fragment of the results for the Analytica sample 'Seat belt safety.ana' are shown in Figure 4.26 for DSI and Figure 4.27 for Analytica.

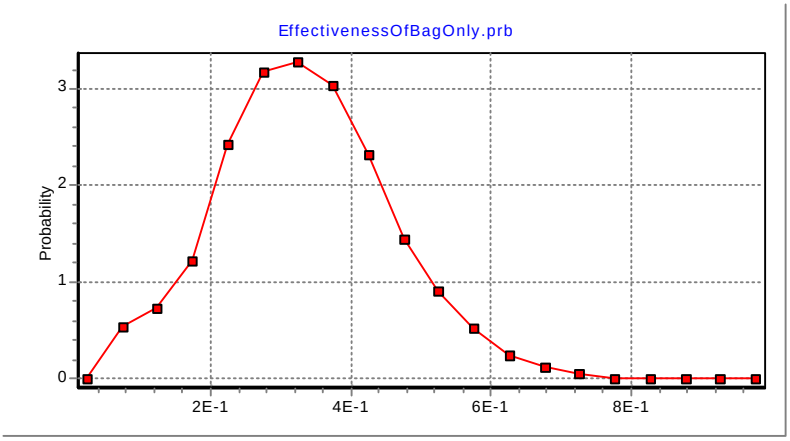


Figure 4.26: DSI results for a fragment of the Seat belt safety sample problem.

The Monte Carlo method inevitably produces artefact spikes due to the random nature of the process, unless many runs are done. In principle Monte Carlo analysis can produce a true representation of the output distribution shape, providing that an algorithm like Latin Hypercube is used[108], but the random spikes can make it difficult to see.

The DSI method consistently produces an output distribution with fewer artefacts and a cleaner shape than the Monte Carlo method.

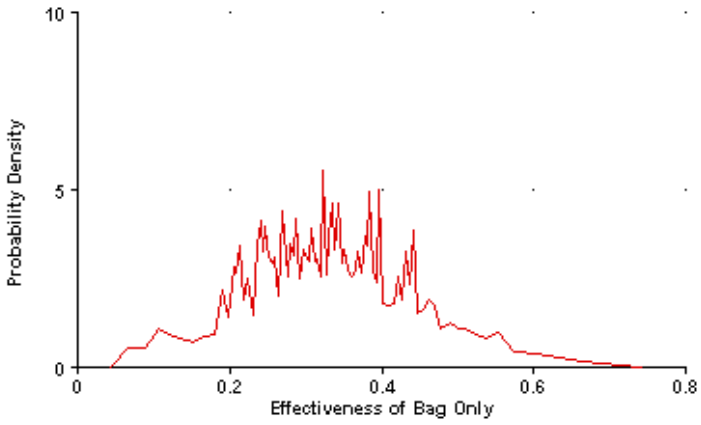


Figure 4.27: Analytica results for a fragment of the 'Seat belt safety.ana' problem.

[108]The classical plain Monte Carlo algorithm is unsuitable if shape of output distribution is required.

Histogram input

Input probability distributions may be defined by histograms in both DSI and Analytica. For example Analytica provides a mechanism for the user to enter a set of cumulative probabilities as fractiles, see Figure 4.28. For example 'Fractiles [.10, .29, .36, .41, .45, .50, .55, .59, .64, .73, .90]' results in the probability density shown here, from Analytica example 'Seat belt safety.ana'. Analytica requires that these must be evenly spaced fractiles. For the Monte Carlo analysis it is not immediately clear whether Analytica uses the data as is, or fits a curve to the data. The latter might be suggested by the extra data spikes in the figure, but this author surmises that it is more likely that these are due to a Monte Carlo process that has already been applied. Analytica shields the user from the Monte Carlo process, so the user has little control or awareness of the process.

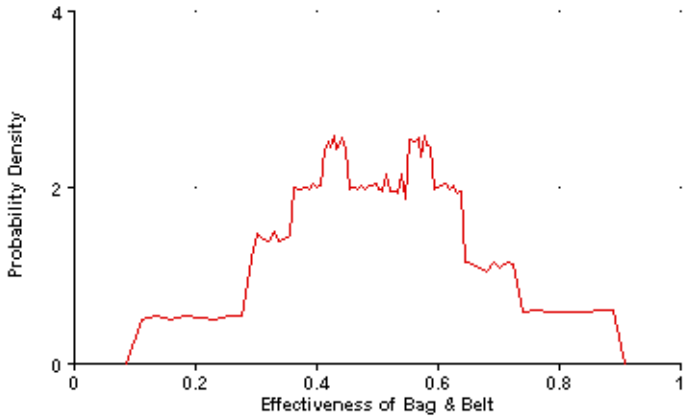


Figure 4.28: A set of cumulative probabilities may be used in Analytica to define a distribution.

DSI can also apply fractiles, and they don't have to be evenly spaced. The Analytica set Fractiles [.10,.29,.36,.41,.45,.50,.55,.59,.64,.73,.90] corresponds to the DSI probability file shown in Table 4.3 and the chart shown in Figure 4.29. The DSI format gives the centre point of the interval, not the upper bound as in Analytica. Therefore the Analytica set corresponds to a DSI origin of 0.10, and first point of (0.10+0.29)/2 = 0.195 and a probability of one tenth.

Both DSI and Monte Carlo methods like Analytica can have histograms as inputs, so they are not differentiated on this aspect.

Title	RelativeEffectivenessOfBagOnly.prb		
genesis	special		
FileVersion	6		
Rows	10		
FileType	N		
DataOrigin	0.1		
Alarms			
0	0.195	0.1	0.1
1	0.325	0.1	0.2
2	0.385	0.1	0.3
3	0.43	0.1	0.4
4	0.475	0.1	0.5
5	0.525	0.1	0.6
6	0.57	0.1	0.7
7	0.615	0.1	0.8
8	0.68	0.1	0.9
9	0.81	0.1	1

Table 4.3: DSI data file.

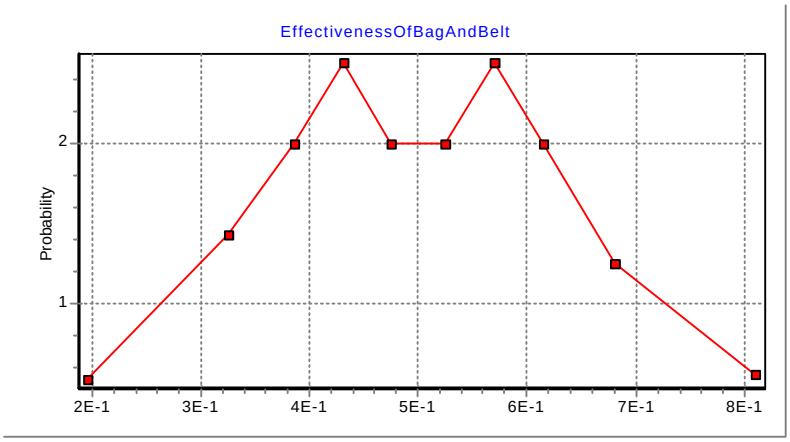


Figure 4.29: The DSI equivalent probability density distribution.

4.6 **Comparison with fuzzy theory**

Consider the case where there are two probability distributions A and B to be added together, as in Figure 4.30. The inputs (light lines) are added together to produce an output (dark curve). This summation was done using the DSI method, though Monte Carlo simulation would provide an equivalent solution. Note that (i) the mode (peak) of the output is not necessarily at the sum of the input modes (due to the effect of the input tails), and (ii) the shape of the output is not triangular like the inputs. In particular, there is not as much weight in the tails of the output as there is in the inputs.

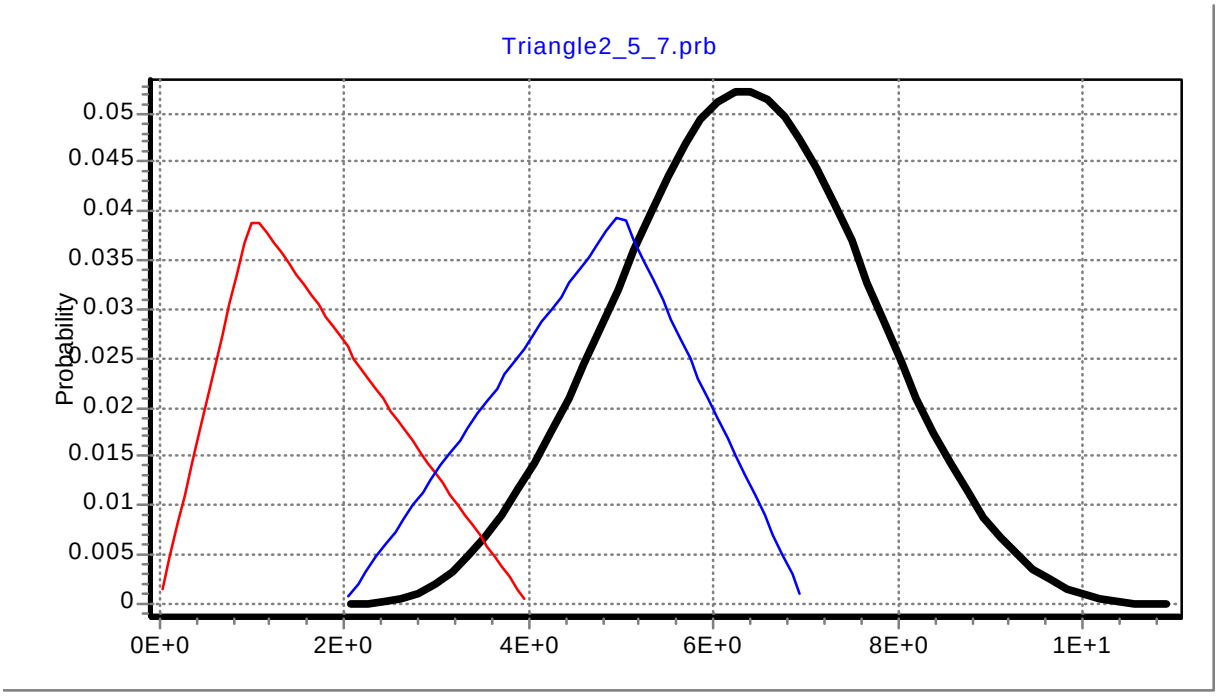


Figure 4.30: DSI result for summation of two triangular probability density distributions.

In the fuzzy theory application the distributions are fuzzified (converted to fuzzy sets), by normalising the probability density function to have a **peak probability** of 1 (in contrast to a probability distribution which is normalised so that the **area under the density** is 1). After fuzzzification the two membership functions can be displayed side by side, and they have the same height. This makes it possible to make an α-level cut as shown in Figure 4.31. Inputs A and B are parameters that vary in the

horizontal (x) dimension (eg physical length of a part) and are represented by membership functions showing how likelihood of occurrence. An arbitrary cut is made at an α-level, producing points $x_L(A,\alpha)$ and $x_R(A,\alpha)$ for A, and $x_L(B,\alpha)$ and $x_R(B,\alpha)$ for B. These may then be used to compute output points on curve C. The α-level is analogous to the confidence limit of a probability distribution. Importantly, the α-level cut is guaranteed to go through both input distributions since the fuzzification ensures they have the same height.

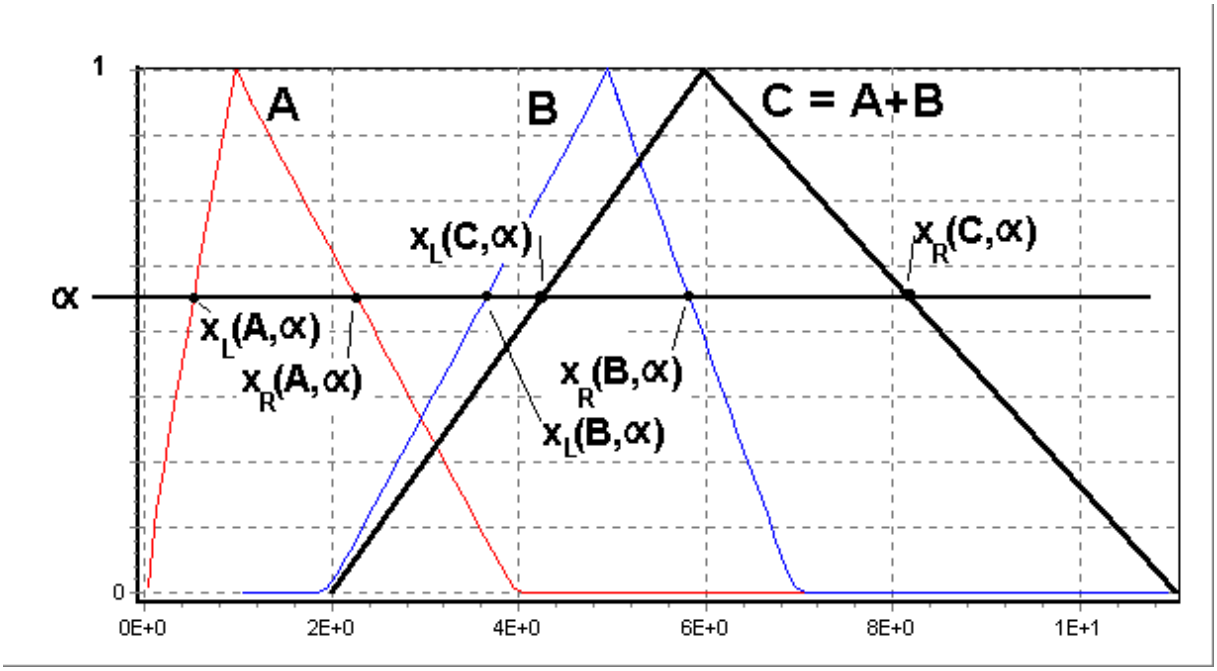


Figure 4.31: Addition of two variable parameters using Fuzzy theory.

The α-level cut produces intersection points $x_L(A,\alpha)$ and $x_R(A,\alpha)$ for A, and $x_L(B,\alpha)$ and $x_R(B,\alpha)$ for B. The fuzzy result C = A + B is then simply computed as the sum of these intersection points: $x_L(C,\alpha) = x_L(A,\alpha) + x_L(B,\alpha)$ and $x_R(C,\alpha) = x_R(A,\alpha) + x_R(B,\alpha)$, which defines two points on the C curve. By repeating the process for other values of α the output curve may be built up point by point (Wood & Antonsson, 1989). The above method involves discrete α-level cuts, and the creation of the output by a composition process. However Fuzzy theory also provides direct solutions for those cases where the membership function is given by an equation.

A valuable side effect of this process is the backward analysis that it provides. This means that for any given α-level on the output C, it is possible to identify the inputs that cause the observed x value of C. This is useful in the design case in that the input needing adjustment can be identified, furthermore the direction of adjustment is also provided (Wood & Antonsson, 1989). It should be noted that this feature requires the storage of intermediate calculations, and is not available by simple inspection of output C.

One of the limitations of the α-level cut method described above is that the membership function must only contain one peak, further, that it may not be concave in the vertical direction. If this condition is not met, then the α-level cut exposes not two but four points on the curve. Realistically there are many cases where the data (eg histogram) violates these requirements, and fuzzy theory is forced to compensate in such situations.

Fuzzy arithmetic works on *interval arithmetic*, as evident in the α-level cut method which only uses the end points, viz $x_L(A,\alpha)$ and $x_R(A,\alpha)$ etc. This provides speedy computation, and makes the backward analysis possible. However Fuzzy theory results are inconsistent with the algebra of random variables,[109] or Monte Carlo analysis, as evident in Figure 4.32. In both cases the inputs are triangular functions (A and B, light lines). The Fuzzy output C = A+B is another triangular function (peak at x=6) whereas the probabilistic computation using DSI gives the dark smooth curve. The fuzzy functions have been normalised for area, for comparison purposes. The vertical axis is histogram probability with fifty bins per distribution.

[109]This may be explained by referring to the *central limit theorem*. A practical example would be the statistical tolerance stack problem in engineering design: when two lengths, each with its own variation, are added together then it is relatively unlikely that the resulting total length would be at one end of the possible range, as this would require that both inputs simultaneously took extreme values. Consequently the probabilistic arithmetic operators of add and subtract give results that approximate a Normal distribution, regardless of the shape of the input distributions. This characteristic is evident in a probabilistic computation such as DSI or Monte Carlo.

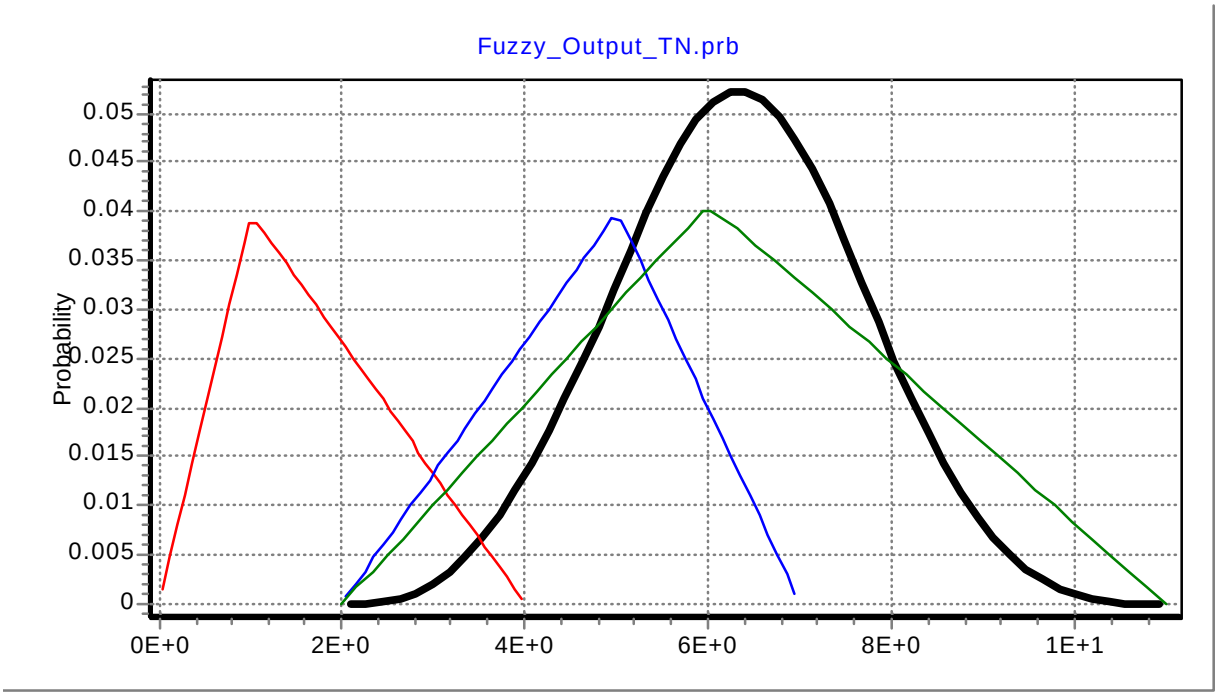


Figure 4.32: Comparison of results for a probabilistic computation and fuzzy theory.

Unlike probabilistic computation methods, the fuzzy result always produces a peak output C at the summation of the two input peaks A and B, regardless of the shape of the input membership functions. An addition of two probability distributions with Monte Carlo or DSI would only show this behaviour if the inputs were symmetrical: if they are skew (as in the figure) then the tails pull the result over towards the tail.

It is observed that many of the fuzzy theory applications use triangular or trapezoidal membership functions. This is because of the ease with which fuzzy computation can be applied to these shapes, for example addition of two triangular functions (a commonly used operator) always produces a triangular output, so only three points need be computed (cf several million for Monte Carlo). However the fuzzy product operator does not have this convenient characteristic, nor do membership functions other than triangular.

Probabilistic computation such as DSI produces superior, even if slower, results than fuzzy theory α-level cuts. See Figure 4.33, which shows the product of two triangular

(0,1,2) functions using probabilistic computation vs fuzzy theory. The horizontal axis is output and the vertical is histogram probability. Both curves have the same vertical scale and may therefore be compared. In both cases the inputs are triangular (0,1,2) functions. The Fuzzy output is the light curve, and the probabilistic computation (using DSI) gives the dark curve. Note that despite the mode of the probability distribution being about 0.7 its mean is 1.0 (not evident from inspection) which is exactly the same as the peak of the fuzzy membership function (which is visible). While the fuzzy output is conservative at the upper range, as evident in a heavier upper tail, it is not conservative at the lower tail. Therefore it cannot be said that fuzzy theory provides a conservative version of probabilistic computation.

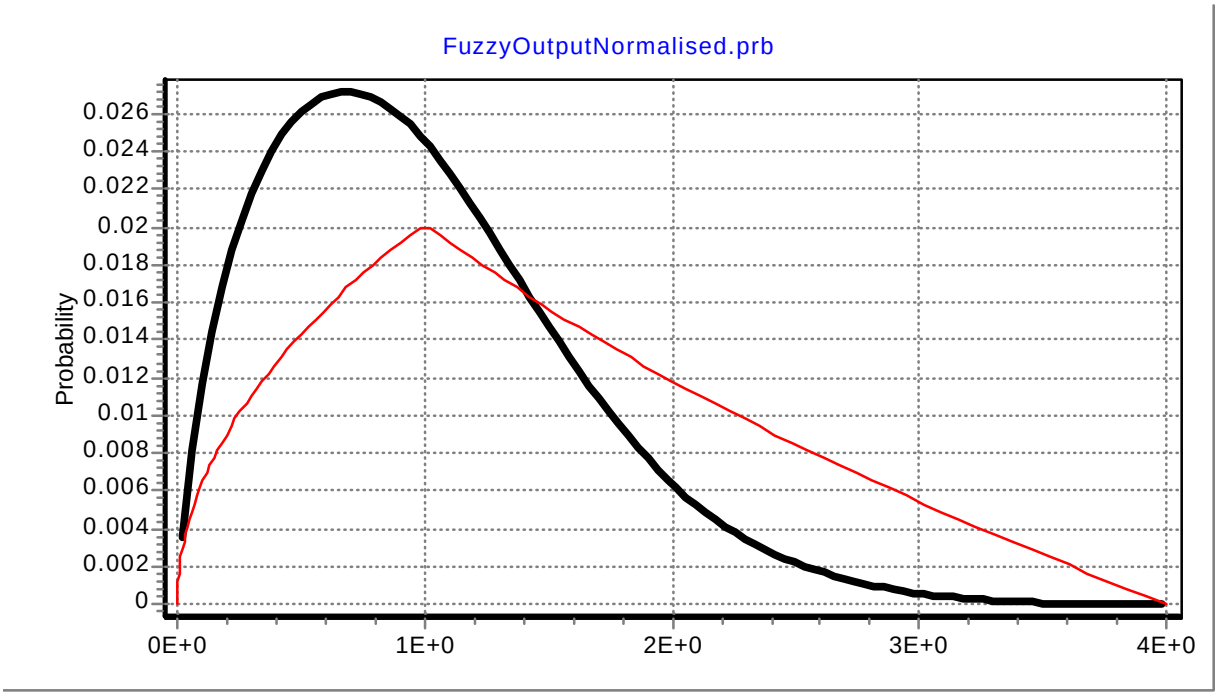


Figure 4.33: Comparison of results for a product operation using probabilistic computation and fuzzy theory.

A significant limitation of fuzzy theory is that it does not scale up to full probabilistic computation. This is easiest explained by considering how fuzzy theory would handle a tolerance stack problem, where two dimensions (each given by a normal distribution derived from empirical data) are to be added to determine the variability of the total length. Fuzzy theory would have to convert the distributions into fuzzy sets. The shape of the input sets would be retained, but in the fuzzy addition the

probability weights would be lost and the output curve would have excess weight in the tails compared to the well established probability calculus. Fuzzy theory would predict the range of outputs, give some indication of the central tendency, but could not be relied on to accurately predict the shape of the output distribution. Therefore fuzzy theory provides only a partial probabilistic computation, and there is no particular merit in providing it with an exact input distribution. Methods other than fuzzy theory provide a better algebra of random variables.

Regrettably it cannot be claimed that fuzzy theory produces a conservative output. All that can be said is that it gives some indication of the range and mean. Where fuzzy theory is superior is in fast computation. DSI produces a slower but more accurate result.

4.7 **Conclusions**

A methodology has been developed to use discrete probabilistic computation in quantitative risk assessment. The method combines two distributions at a time with a mathematical operator, and produces an output distribution. In principle any type of probability distribution may be modelled, and the inputs needs not be of the same type.

The method has been implemented in software, using the Delphi programming language, and provides a stand-alone executable that runs under Microsoft Windows operating systems. A variety of common mathematical operators (arithmetic and logical) have been demonstrated, and the principles are extendable to other operators. The existing suit of operators may be used to model various relationships such as function, performance, reliability, dimensional tolerances, cost, etc, in the quantitative domain.

Compared to the histogram Monte Carlo method, DSI produces results that are consistent with Monte Carlo, but quicker, without the uncertainty of convergence, and with less artefacts.

Compared to Fuzzy theory α-level cuts, DSI produces more accurate results though slower. It does not have the backward analysis capability of fuzzy theory.

The DSI software has also been provided with both Monte Carlo and Fuzzy cut set algorithms. The user may therefore select which computational tool to use, and even use multiple tools in one model if that is meaningful.

Chapter 5

Qualitative probabilistic computation

This chapter documents the method used to describe functional relationships in qualitative (textual) terms. The mechanism permits an expert to express confidence in those relationships. The method is applicable to modelling early design decisions where information is sparse and qualitative.

5.1 Introduction

The move to shorter time-to-market and concurrent engineering philosophies has put pressure on the design function. In particular there is a need to ensure early in the design process that functional and other integrity issues are achieved and not compromised by subsequent design changes. The challenge is to develop methodologies that can operate in the early design stages where information is sparse and uncertainty is high. This chapter describes the development and implementation of such a methodology.

The term '*uncertainty'* has flexible usage in the literature. It is therefore necessary to clarify how the term is used in this thesis. The distinction is made between two types of uncertainty: the *analysis uncertainty*, and the *process variability.*

5.1.1 Analysis uncertainty

System relationships that can be quantified, i.e. expressed mathematically, and which operate on quantitative data, have no *analysis uncertainty* (or *uncertainty of analysis*). However there are many cases where the fundamental relationships describing a process are qualitative,[110] i.e. uncertain and subjective.

Qualitative relationships are expressed not in mathematical statements, but in rules, guidelines, and subjective statements of text. They operate on qualitative or quantitative data. The mere existence of quantitative input data does not necessarily mean that relationships are quantitative. The concept of analysis uncertainty is compatible with subjective probability as defined by Lad (1996).[111]

---

[110]The interested reader is referred to Chapter 6 for a description of how the terms quantitative and qualitative are used in this work.

[111]Lad (1996) rejects the viewpoint that a measurement yields a true value plus a measurement error, i.e. that the true value is unobservable. He asserts this is operationally meaningless. Instead he asserts that any measurement is only the numerical outcome of a specified procedure. He describes *prevision* as the measurement (using an operational procedure) of an individual's uncertain knowledge. It is a probability, and it can be specified by ranking it within an order

5.1.2 **Process variability**

Whereas analysis uncertainty refers to the uncertainty of the relationships, *process variability* refers to the uncertainty of the data. The process variability may be well known, for example, it might be known that water temperature in a dishwasher is 65^{o}C mean with standard deviation 5^{o}C. The process variability is usually easy to quantify, given sufficient data observations, and it takes the form of a probability distribution of one kind or another, perhaps the Normal distribution. Process variability can also take the form of *reliability,* which is a measure of the probability of failure with time, and is often represented by a Weibull probability distribution.

Process variability can exist independently of uncertainty of analysis. For example there is no uncertainty of analysis in a relationship like F = m.a. Given two inputs (m and a), there is only one output (F) that is possible. However there could be process variability in the inputs. For example both *m* and *a* could vary. Physical interpretations could be process variability in the *mass* of a manufactured object, and variability in the *acceleration* that it will see in use (an environmental variability). The *force* outcome will therefore have a process variability that it inherits from its inputs.

Process variability also includes reliability and other parameters that show variation over time (rather than over sample). In both cases the parameter may be represented by probability distribution. There is a third source of variability, which is noise on a signal, but this is not relevant to the present discussion.

---

based on the individual's opinions and utility valuations. An order system could be a set of betting comparisons.

5.1.3 Exploring the wash performance issue

The approach taken in the present work is to rely on the human designer for the creative input to the design, but to develop a methodology to assist the human designer or design manager in the decision making process and the management of design. The complicating factor is the *uncertainty of analysis*. For example it might be known that dishwasher wash performance improves with higher water temperature, though the precise relationship is unknown, i.e. there is large uncertainty in the analysis. The little that is known is qualitative, and unsuitable for processing through a conventional quantitative simulation system.

In this study the wash performance of dishwashers falls into the qualitative category. The critical question is:

How can qualitative behaviour (such as wash performance) be simulated when there is high uncertainty of analysis?

This is not an easy question to answer. The issue is usually ignored by typical engineering analysis tools that seek to apply functional modelling. They generally do not attempt to quantify the uncertainty let alone factor it into the simulation. Instead they leave it to the judgement of a human expert to deal with the many ambiguities.

At the top level, there is the desire to have a prediction of wash performance, with not only the predicted value, but also the probability of that value. Ideally there should be a continuous rather than a discrete probability distribution. In the case of wash performance it is known that water temperature provides process variability that will cause some spread in wash performance. Further spread in predicted wash performance will be caused by the uncertainty of the analysis, but just how much more spread needs to be determined. Process variability is not the main issue here,

as there exist tools to quantify it. Instead the main issue is analysis uncertainty, eg just how water temperature affects wash performance.

A common philosophy behind the scientific paradigm is that any observed behaviour, however perplexing, may ultimately be quantified and expressed formally. From this follows that qualitative relationships arise not because a system is fundamentally undecipherable, but rather because there is not yet sufficient knowledge of the system to be able to quantify it. With sufficient effort, any qualitative system may be converted into a quantitative one. Naturally it is not always practical to research a system to the necessary depth to be able to quantify the system relationships. Commercial engineering product development does not have the resources to achieve this, perhaps largely because the costs of fully quantifying something like wash performance might be expected to exceed the benefits. Consequently, engineering product development often needs to proceed on the basis of qualitative relationships.

Qualitative relationships refer to uncertainty of analysis and are therefore subjective. Therefore, to address the issue of predicting performance in a systems, it is necessary to take into account the subjectivity of beliefs. If a relationship can be expressed in a qualitative system, then it involves a degree of subjectivity or belief. It requires the subjective opinion of an expert, and naturally other experts may express the relationship differently. There always remains an element of uncertainty about the relationship.

For example, let us assume that an expert person is of the opinion that *wash performance* is determined both by *soil removal* and by *rinse effectiveness*. Furthermore, the expert might be able to express some general principles about the relationships between the two. However those relationships will be somewhat vague. If another expert were canvassed, then it is entirely possible that a different set of relationships would emerge.

There are two primary parts to the qualitative modelling problem:

(1) how to express such relationships

(2) how to simulate system behaviour from these relationships

Additionally, there is a third problem, which is:

(3) how to combine quantitative and qualitative relationships.

The first two issues are addressed in this chapter, and the third in the next. There exists a fourth issue, which is not addressed at all, namely that qualitative relationships necessarily express the opinion of an expert, and it might be desirable to incorporate the probability of different opinions as to how those relationships are constructed, i.e. to incorporate some of the team issues.

5.1.4 Existing approaches to uncertainty of analysis

Conventional engineering analysis tools are predominately deterministic as they produce no indication of the uncertainty of the output. *Monte Carlo* analysis may be used to overcome this, providing the inputs are quantitative and there is no uncertainty of analysis. Neither do *functional modelling* systems accommodate process variablity let alone uncertainty of analysis (there are partial exceptions such as Crossland et al (1995) [112]). However none accommodate uncertainty of analysis. There are functional modelling systems based on natural language fragments (eg Deng et al 1998), but these likewise do not support the modelling of uncertainty of analysis.

[112] Crossland et al (1995) address the issue of uncertainty in the early design stages of design. They propose using a shared object-oriented model of a design, with the intent that both the level of detail and the level of certainty can be changed as the design progresses so that the proposed system "is able to represent uncertain relationships between objects thus permitting modelling of alternative (parallel) design paths, as well as uncertain attribute values".

Various *artificial intelligence* methods such as *expert systems*[113] have been used. Expert systems sometimes accommodate an uncertainty factor but this is used in a loose manner, and not always with a thorough statistical interpretation. Other expert systems have used *fuzzy theory* to determine the extent to which a rule applies. However neither of these approaches adequately addresses the modelling of uncertainty of analysis, and there remains also the problem that expert systems have generally only been successful in well-defined applications that have fixed design strategies (Tang, 1996; Bartsch-Sporl and Bakhtari, 1996).[114]

The *Bayesian* methods of probabilistic reasoning have also been used to model quantitative uncertainty. The uncertainty has to be described explicitly by a probability distribution, and Siu (1990) notes that this a burdensome process and that it is difficult to find a formal solution because of the large number of variables and the uncertainty in the data. Also Bayesian methods do not support qualitative relationships.

Quality function deployment (QFD) is a technique that seeks to distill product attributes out of loosely defined customer requirements using scores. The process is therefore analogous to the conversion of subjective qualitative relationships into Numerical qualitative relationships.[115] Initial scores are subjective, but propagate

---

[113]Expert systems describe the system relationships in terms of rules, which are responses to given conditions, much like *if..then* statements. The main difference between expert systems and conventional proceedural programming languages is that expert systems have an engine which determines whether a rule statement should be executed, whereas proceedural programming languages control program flow explicitly with code.

[114]Bartsch-Sporl and Bakhtari (1996) set out the requirements of artificial intelligence implementations as:

- Completeness of knowledge in the domain model
- Consistent logic within the domain
- Closed domain, not vulnerable to outside effects
- Problems that can be decomposed and re-composed
- Solution spaces that can be formally defined.

They point out that real life design domains seldom meet these requirements precisely, and their strategy has been to use artificial intelligence not so much to solve design problems on its own, but rather to assist the human designer.

[115]The QFD method assigns a numerical score to the relative importance of the customer requirements. This score represents the opinion of those setting up the analysis and is therefore entirely subjective and belief based. Perhaps more significantly, the scoring system imposes a numerical rank on the customer requirements. These scores are then propagated through the matrix to

through to the results, and are therefore significant. Moreover, the QFD method has no means to accommodate uncertainty in either the scoring or the relationships. The somewhat related methods of *conjoint analysis* and the *analytical hierarchy process* (AHP) are also used on subjective data. They produce a rank for such data, which can be useful in prioritising customer preferences, but the systems are not simulation systems and neither do they accommodate uncertainty.

The *fuzzy theory* approach is to convert the qualitative scale into a numerical one, and then apply mathematical operators to it. This is also the approach taken in *QFD*, where customer requirements are given a weighting factor. The *design structure matrix* approach also uses this method. However the weakness of these approaches is that the weighting is necessarily arbitrary. Also, they operate best where the variables can be ranked, eg they can cope with a variable like *cold - warm - hot* since they assign a number or range of numbers to each of *cold* etc. However they find it more difficult to deal with variables lacking rank, such as food soil of *sauce - rice - eggs - oats*. This is the weak order problem discussed by Scott and Antonsson (1999). See Chapter 6 for further discussion on this important topic.

Possibly the most comprehensive approach to modelling uncertainty in qualitative parameters is *decision analysis.* This method is specifically structured to accommodate uncertainty of analysis, since this is necessary when analysing risk. As the name suggests, decision analysis is typically applied to problems involving decisions, though it simulates outcomes rather than making a decision on behalf of the user.

---

determine the engineering characteristics of the product.

5.1.5 Examples of decision analysis

The way that a decision analysis system like *Hugin* [116] approaches qualitative analysis is illustrated with an example. A second example is given for *Analytica*.

Hugin example

This models the possible (failure) events of a fill valve. This device is used in dishwashers and similar systems to let water into the machine from the domestic water supply. The *fill valve* should permit *water flow*, though there are several ways in which it could fail. These are *none* (no water flow at all), *ok* (functions as intended), or *leak* (water leaks from the device). Lower level events are shown in Figure 5.1 as *water pressure* (*none, low, ok, high*) and *Command* from the controller (*none, ok, stray signal*).

Each of these nodes can be set up in *Hugin*, with the possible states for each. The decision analysis software automatically constructs a matrix which the user must fill in. This matrix is also shown in the figure. The total number of elements in the matrix is the product of all the states involved, in this case 4x3x3 = 36. Space prevents them from all being shown in the figure.

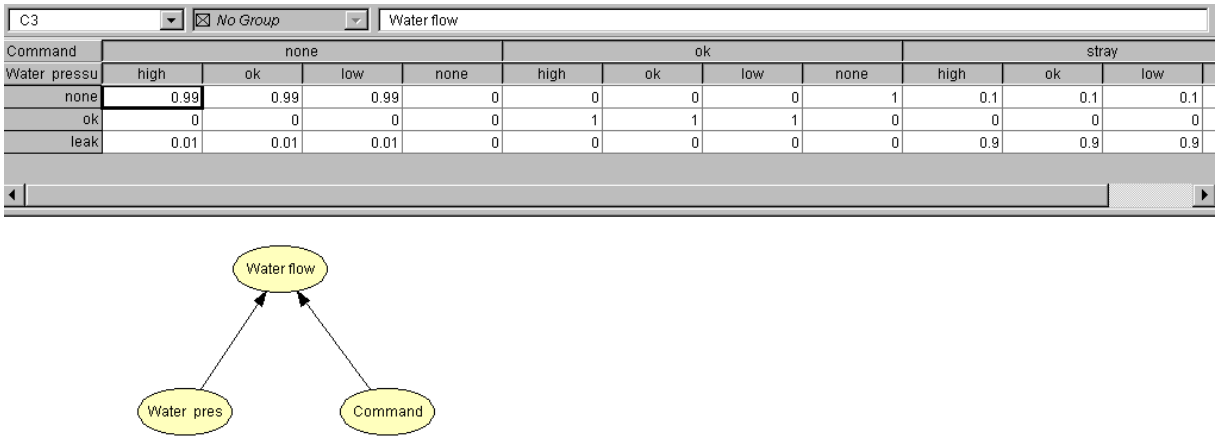


Figure 5.1: Decision analysis applied with Hugin.

The user has to provide the probability for each element. For example, the first element is the probability of getting *Water flow - none*, given *Command - none*, and

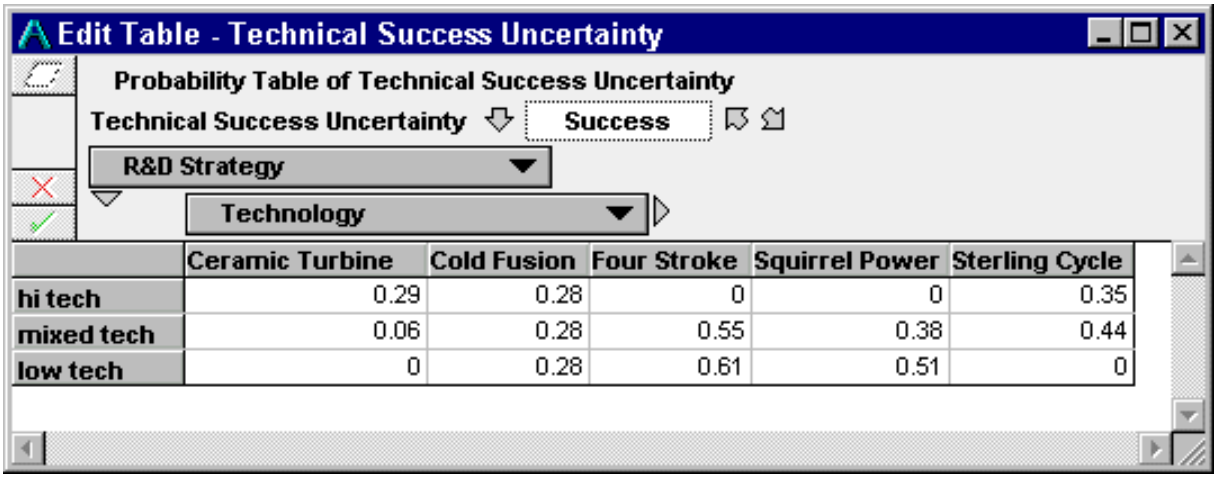
[116] Hugin is decision analysis software from http://www.hugin.dk/.

Water pressure - high. In this case the user has assessed the probability as 0.99. In other words, there is unlikely to be water flow from the fill valve if the opening command is absent, even if the water pressure is high.

Decision analysis uses a *combinatorial* approach, as every state is combined with every other state, a probability assigned, and the result entered into the table. A significant disadvantage of this approach is the large number of entries that have to be made when there are many states. These assessments are made manually, and therefore tend to limit the application to cases of few states. Decisions usually involve only two or three choices at each stage, and this limits their usefulness where a large number of outcomes are modelled.

Analytica example

Analytica is another decision analysis system. It provides a 'probability table' to describe a variable as a discrete uncertainty. The variable may be a list of numbers or of labels. For example, the model 'LEV R&D Strategy.ana' provided with Analytica uses a probability table as shown in Figure 5.2.



	Ceramic Turbine	Cold Fusion	Four Stroke	Squirrel Power	Sterling Cycle
hi tech	0.29	0.28	0	0	0.35
mixed tech	0.06	0.28	0.55	0.38	0.44
low tech	0	0.28	0.61	0.51	0

Figure 5.2: Probability table from a sample file in Analytica. The 'Technical Success Uncertainty' is the output and it may be 'Success' or 'Failure'. Only the 'Success' slice of the table is shown here. The 'R&D Strategy' may be 'hi tech', 'mixed tech' or 'low tech'. The 'Technology' may be 'Ceramic Turbine' etc. The table gives the probability of 'Success' given inputs for 'R&D Strategy' and 'Technology'. For example, 'hi tech' and 'Ceramic Turbine' have a 0.29 probability of success (and a 0.71 probability of failure, not shown).

Analytica also provides that the user can assert a qualitative distribution (eg weather is 0.4 sunny and 0.6 rainy) as a one-dimensional probability table. This may then be fed into the model and processed further. The system is therefore similar to decision tables.

5.2 **Probabilistic computation of maps**

The decision table approach appears to be the most suitable methodology of processing qualitative data and representing qualitative relationships. The solution followed was to use subjective probability (belief of an expert) to model diffuse relationships.

5.2.1 **Description of the method**

The principles of decision analysis were applied to create a system that could simulate qualitative performance of a system, and propagate probability at the same time. This section provides an example of the decision table or *map* relationship, exploring the relationship of Figure 5.3.

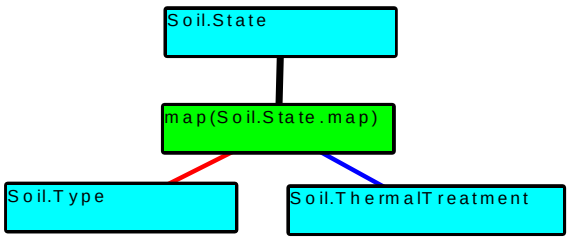


Figure 5.3: View of map operator in Design for System Integrity (DSI) software, showing that two inputs, Soil type and Soil thermal treatment are used to determine Soil state.

While quantitative relationships give rise to numerical relationships and are defined in terms of mathematical operators, qualitative relationships are defined in textual terms. These could be pseudo-scalar values such as a temperature scale (*hot, warm, cool, cold*) or abstract terms such as failure modes for an electric motor (*open-circuit, short-circuit, overheat, normal*). They can also be numbers that are interpreted as text strings (eg *10%, 20%, 50%*).

Maps make up a significant part of a subjective model such as that for wash performance of dishwashers. This is to be expected given the lack of quantitative understanding about the fundamental wash mechanisms.

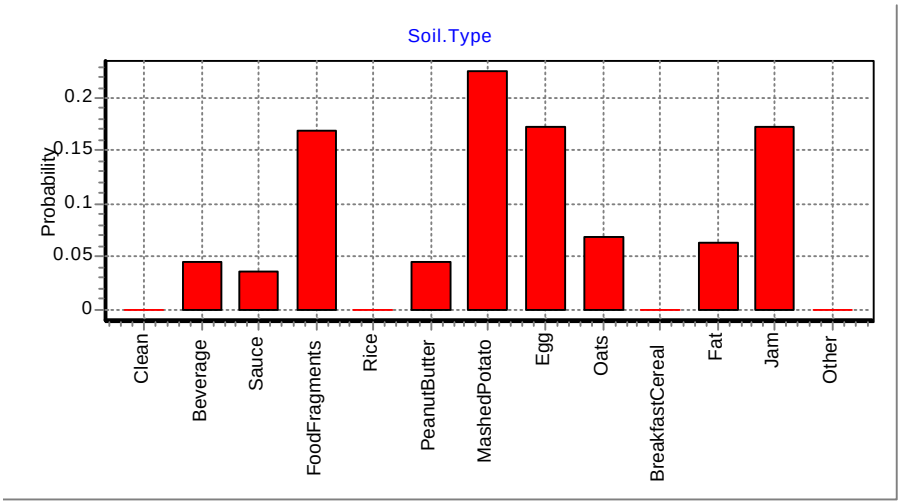


Figure 5.4: Soil type shown as a histogram based on mass of various soils used in the ANSI/AHAM test.

In this example the soil type is to be combined with the soil thermal treatment to produce the soil state. Each of these parameters is an array of qualitative text descriptors with a numerical probability.

Soil type

Soil refers to the type of foodstuff on the dishware. A typical soil profile is shown in Figure 5.4.

Soil thermal treatment

The soil on the dishware could have various thermal treatments, such as *fresh, dried-on, reheated, baked*, or *burned*. These are shown in

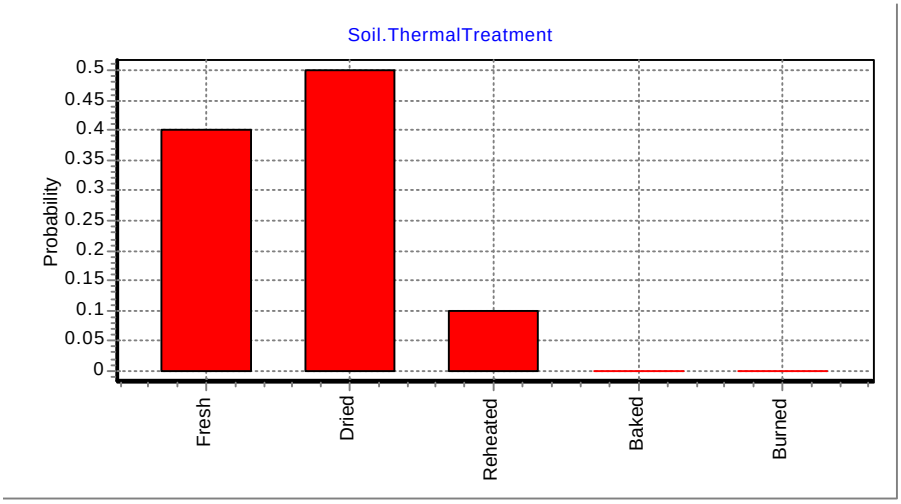


Figure 5.5: The profile for Soil thermal treatment can range from 'fresh' to 'burned' soil. For the ANSI/AHAM test there is only a single value: 'dried'. In this particular example it has also been asserted that some of the soil is 'fresh' and some 'reheated'.

Figure 5.5, where a probability for each has been asserted.

Soil State

This parameter is computed as a subjective map between soil type and soil thermal treatment. A fragment of this map is shown in Figure 5.6. The map process corresponds to a decision table. For example, if the soil type is '*Rice*' and the soil thermal treatment is '*Fresh*', then the expert is sure (probability 1) that the soil state will be '*LoosePieces*'. This approach also permits the inclusion of uncertainty, for example with soil type '*Egg*' and soil thermal treatment '*Fresh*', then the soil type is expected to be '*SolubleFilm*' and '*StickyPaste*' in equal probabilities (0.5). Similarly other more extended applications of this principle are evident in the figure, where the expert has split the probability over several outcomes.

Unlike some other methods (such as *fuzzy theory* and *QFD*) the outcomes themselves are still qualitative text descriptors, and they are not converted to any numerical scale or even ranked. Nor is any mathematical operator applied. All that is quantified is the probability. When the expert is sure of the outcome, then he assigns a probability of 1 to it. When less sure, he splits the probability over several outcomes according to how he believes them likely. The figure shows the portion of the map for '*Fresh*', and the principles apply to the other soil thermal treatments though it is not practical to shown them all here.

Soil.State			Soil.Type												
			Clean	Bever	Sauce	FoodF	Rice	Peanut	Mashe	Egg	Oats	Break	Fat	Jam	Other
Soil.Therm	Fresh	Comment													
		Clean	1	0	0	0	0	0	0	0	0	0	0	0	0
		SolubleFilm	0	0.333	0.1	0	0	0	0	0.5	0	0	0	0.5	0.1428
		FineParticulate	0	0.333	0.2	0.25	0	0	1	0	0.333	0.333	0	0	0.1428
		LoosePieces	0	0	0.2	0.5	1	0	0	0	0.333	0.333	0	0	0.1428
		StickyPaste	0	0	0.3	0.25	0	1	0	0.5	0.333	0.333	0	0.5	0.1428
		Greasy	0	0.333	0.2	0	0	0	0	0	0	0	1	0	0.1428
		DriedStuck	0	0	0	0	0	0	0	0	0	0	0	0	0.1428
		CookedOn	0	0	0	0	0	0	0	0	0	0	0	0	0.1428
		BurnedOn	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5.6: Map to convert soil type and soil thermal treatment into soil state. Numbers are probabilities where any one column will sum to unity.

The inputs '*soil type*' and '*soil thermal treatment*' have probabilities that are asserted independently of the map. Previous figures 5.4 and 5.5 have illustrated these assertions, but it is important to note that the user can modify the profiles of these assertions. Other soil profiles may easily be set up, for example one that includes more *rice*, so that performance of the machine in a different market may be simulated. In changing the assertion the user is reassigning the probabilities of the assertions only, and the map probabilities remain fixed regardless of the asserted profile. Of course the map probabilities MAY be changed, but that corresponds to the expert changing his mind about the model. The map will have to be edited if one of the inputs gets a new qualitative item, for example if a new soil type (eg '*honey*') was added.[117]

The assert probabilities and the map probabilities are joined in a combinatorial fashion to determine the output probabilities for '*soil state*'. For example the probability of '*sauce*' has been asserted as 0.0370, and the probability of '*fresh*' as 0.4000. The joint probability of these two events occurring is the product of their

[117] If the map were not edited then the software would fail to recognise the new soil type and any probability asserted to that soil type would be assigned to an '*unresolved*' category of soil state.

individual probabilities, i.e. 0.0370x0.4000 = 0.0148 (*independence* assumed). Using the map, and entering at column '*sauce*' and row set '*fresh*', the map outcomes and their probabilities are given as *SolubleFilm* (0.1), *FineParticles* (0.2), *LoosePieces* (0.2), *StickyPaste* (0.3), *Greasy* (0.2).

Now the joint probability must be split across all these outcomes, which is a straightforward multiplication, giving *SolubleFilm* (0.1 x 0.0148 = 0.00148), *FineParticles* (0.2 x 0.0148 = 0.00296), *LoosePieces* (0.2 x 0.0148 = 0.00296), *StickyPaste* (0.3 x 0.0148 = 0.00444), *Greasy* (0.2 x 0.0148 = 0.00296). This process is repeated combinatorially for every soil type and soil thermal treatment, and at each combination step the outcomes (eg *SolubleFilm*) have an opportunity to pick up additional probability contributions, which are added to that which they already hold. At the end of the process each of the outcome descriptors will have a probability associated with it, and these can be displayed as a chart. See Figure 5.7 for the output.

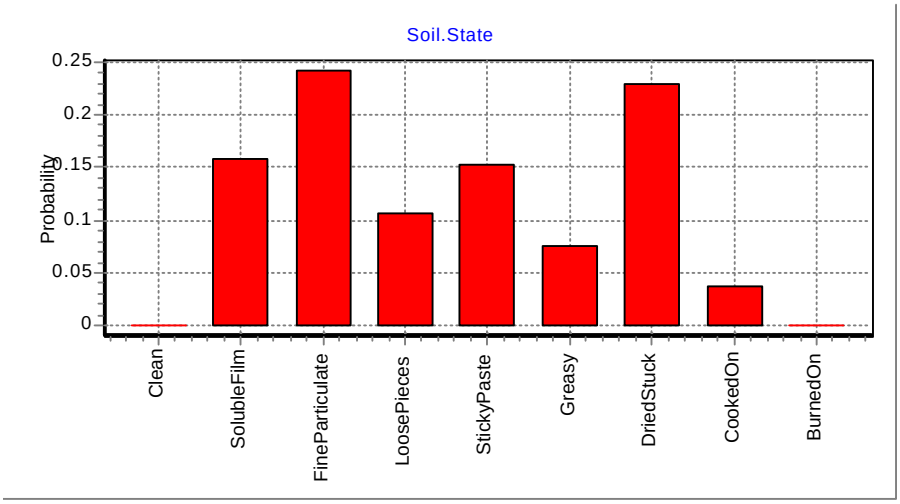


Figure 5.7: Soil state is a combination of soil type and soil thermal treatment. It describes the important wash characteristics of the soil.

Though the map may be edited, in a stable model it should be a static structure. However the inputs will vary as different assertions are made and propagated through the map. An important feature of the map process is that it places no

numeracy constraints on the inputs, for example ‘*Clean, SolubleFilm, FineParticulate*..’ are only qualitative. Neither do the inputs have to be in a ranked scale. These characteristics differentiate the method from fuzzy theory and QFD. The map accommodates qualitative inputs by treating the inputs purely as text strings.

5.2.2 **Implementation in software**

The map process has been implemented in software using the *Delphi*[118] programming language. The finished application is called Integrity-T, and though it can operate on its own, its intended use is tightly integrated with the Integrity-N quantitative (numerical) probability simulation tool to make up a bigger software application called Design for System Integrity (DSI).

Implementing the plain qualitative Integrity-T involved providing the user with grids in which the two input assert files may be created, and another grid for the output data. Charts are also provided to visualise the data. Another grid is provided for the map. The software provides for the map to be created manually or semi-automatically in the software, or imported from a spreadsheet. The recommended process for creating the map is first to create the text string labels for the two input and one output arrays (the probabilities are immaterial at this stage). Then a simple button click creates a map using those labels, correctly formatted and ready to receive the expert beliefs. One the map is created then the input probabilities may be asserted and the simulation run.

The primary functionality of the map is based on decision tables and no academic novelty is being claimed in that regard in this project. Where the map process departs from conventional decision tables is in its ability to accept quantitative inputs too, and

[118]Delphi provides a blank form and various components such as buttons and grids. The application was built around such components, with the author creating the necessary code to perform the mathematical processes, file handling, and display of results. Further details are provided in a subsequent chapter.

this is described in the next chapter. This map process permits any combination of textual and numerical relationships to be defined.

5.2.3 Applications for qualitative maps

It is suggested that the qualitative map process be used in two stages. First an expert would create the model and the map, using his belief of the processes being modelled. Thereafter a user, who need not be the expert, could assert input probabilities and propagate them through the map to determine the outputs. This user would not need to edit the map or even understand the expert's thinking.

The intent is that a change in a design parameter (eg at concept, embodiment or detail stage) will propagate through the system. This means that the risk in the result may be identified, by quantifying the probability of good and bad outcomes. Importantly, the qualitative probabilistic computation developed here is able to process both the uncertainty of analysis (probability array within the map) as well as propagating process variability (probability arrays of the assertions) through the model to determine qualitative outputs with their probabilities. In the early stages of design, all the probability distributions will tend to be broader and will propagate more uncertainty through the model. Reduction of uncertainty occurs as the design progresses to more concrete stages and more prototype test results become available. This results in greater certainty about functional relationships and less process variability, both of which result in narrower distributions of final values. The combined effect of the probability distributions on values of a large number of parameters can be determined to show the overall robustness of the design.

5.3 **Validation**

For validation a sample output from the DSI textual method is checked against known results.

The sample problem is shown in Figure 5.8 as a graph, being the determination of *Weather* from *Season* and *Wind* using a map. The map is given in Table 5.1.

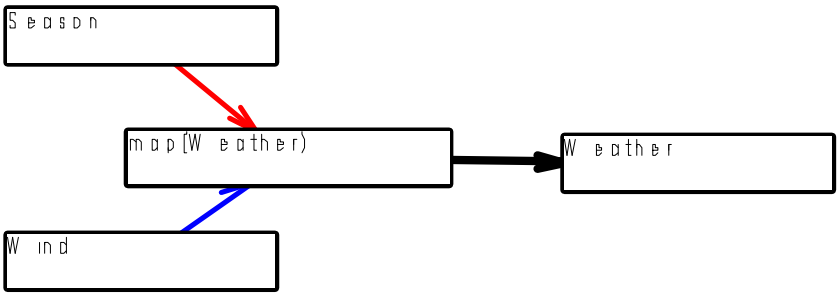


Figure 5.8: Graph for determining Weather from Season and Wind.

Weather			Season			
			Summer	Autumn	Winter	Spring
Wind	North	Comment				
		Wet	0	0	0	0
		Mild	0	0.3	1	1
		Hot	1	0.7	0	0
	South	Comment				
		Wet	0.4	0.7	1	1
		Mild	0.6	0.3	0	0
		Hot	0	0	0	0
	None	Comment				
		Wet	0	0.5	0	0
		Mild	1	0.5	1	1
		Hot	0	0	0	0

Table 5.1 Map for determining Weather from Season and Wind.

For given *Season* (eg Autumn) and *Wind* (eg North) the map shows the outputs and their probabilities (see highlighted cells in table, i.e. *Weather* Mild 0.3, and Hot 0.7). The probabilities in this map are simply for demonstration purposes.

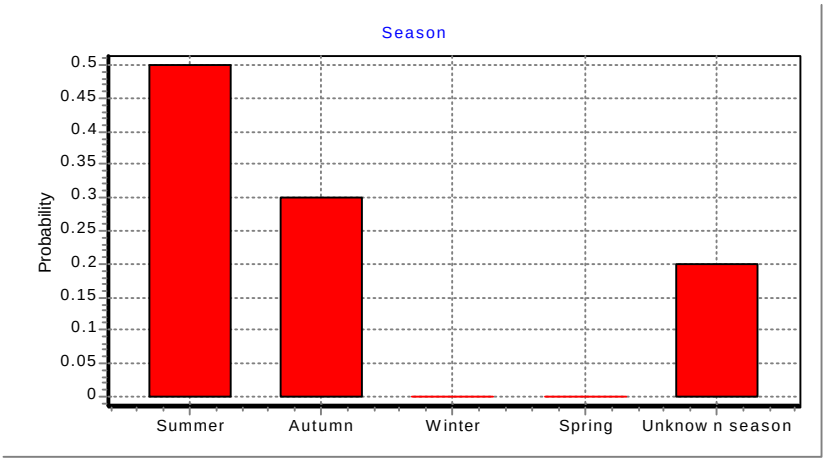


Figure 5.9: Asserted probability for qualitative parameter Season.

For the validation case the *Season* is asserted to be as shown in Figure 5.9, and the *Wind* as in Figure 5.10. In both cases there are additional labels (*unknown season*, and *unknown wind* respectively) that do not appear in the map, since part of the validation is to check how these disturbances are propagated through the system.

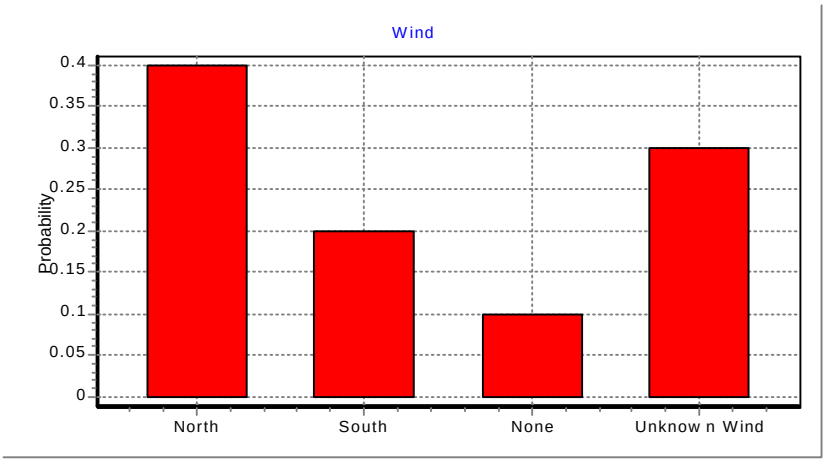


Figure 5.10: Asserted probability for qualitative parameter Wind.

The theoretical output of the system may be readily calculated. For example the probability of *wet* weather is determined as the sum of the product of the assert probabilities and the respective map probability. Thus:

Probability(wet) = 0.5x0.2x0.4 + 0.3x0.2x0.7 + 0.3x0.1x0.5 = 0.0970

Probability(mild) = 0.5x0.2x0.6 + 0.5x0.1x1.0 + 0.3x0.4x0.3 + 0.3x0.2x0.3 + 0.3x0.1x0.5 = 0.1790

Probability(hot) = 0.5x0.4x1 + 0.3x0.4x0.7 = 0.2840

Probability(unresolved) = 0.2x0.4x1 + 0.2x0.2x1 + 0.2x0.1x1 + 0.5x0.3x0.1 + 0.3x0.3x1 + 0.2x0.3x1 = 0.4400

The unresolved probability need not specifically be represented in the map, as the software recognises it as deviant and automatically creates an *unresolved* label in the output. The value of the unresolved probability is determined using an implicit map probability of unity, eg. if the *Season* is *unknown season* then the output is automatically *Weather unresolved* regardless of the *Wind*.

DSI produces exactly the same results as the above theoretical calculations, as evident in Figure 5.11.

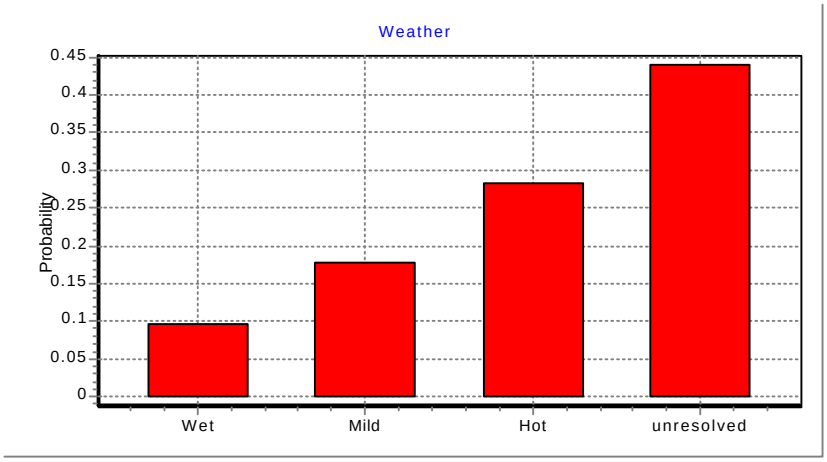


Figure 5.11: Calculated probability for Weather.

This set of results demonstrates that DSI performs as required for the test case. It produces the correct output labels and their probabilities. Furthermore it automatically creates an unresolved label in the output (only where necessary) when one or both inputs contain labels that were not anticipated in the original map. It correctly assigns probability to this unresolved label.[119]

Larger maps operate on identical principles, simply with larger arrays. It is therefore reasonable to assume that the method is valid for all maps.

[119] If this result containing an *unresolved* element were propagated to another higher level map, then the another *unresolved* label would be generated from that second map. Thus the *unresolved* probability eventually appears in the final result.

5.4 **Conclusions**

At early design there is considerable uncertainty in the functional model of the product and indeed in any simulation model. Some of this uncertainty is the randomness of the input variables, i.e. process variability. Most simulation systems as used in functional modelling are unable to accommodate process variability in the model. This constraint also applies to the conventional engineering analysis and computational tools which are predominately deterministic in nature (i.e. produce a single output number with no indication of the uncertainty band around it). A few methods are available to handle process variability, namely the algebra of random variables, Monte Carlo analysis, and fuzzy theory.

However the uncertainty of analysis at early design often defeats even these tools. This is because the listed tools utilise algebraic operators internally, and therefore require quantitative input variables, or at least qualitative variables that can easily be converted to a numerical scale. Unfortunately not all variables at early design can be quantified.

A method has been demonstrated whereby qualitative relationships may be modelled using a decision table approach which has been called the Integrity-T map process. The system has been implemented in software. This permits both the uncertainty of analysis and the process variability to be modelled and propagated through the simulation system. The output of the system is an array of qualitative terms (text string labels) together with the probability of each element.

Importantly, the map process places no numeracy constraints on the inputs, nor even ranking, and this differentiates the method from fuzzy theory and QFD. These other methods are restrictive in that they convert a qualitative variable into a number and then process that number algebraically. The map process avoids this restriction by directly *correlating* input variables to an output variable using confidence level

estimates. In such a process only the confidence level estimates need to be quantitative (numerical).

Qualitative relationships are based on the underlying beliefs of the expert. The map encapsulates the belief of the expert as to how the parameters affect the outcome. The method does not insist on the existence of any explicit rationale for these beliefs. It is important to note that the qualitative Integrity-T method does NOT require that an expert be certain in his beliefs. Instead it accommodates vagueness of belief as a type of process variability and propagates it through the model.

The method is flexible enough to accommodate the diffuse information, and even subjective opinion, typical of the early design stages. This permits the risk and uncertainty in a simulation output to be determined even at early design.

Chapter 6

Combining qualitative and quantitative probabilistic computation

The Design for System Integrity (DSI) methodology permits the use of qualitative and quantitative relationships in one probability simulation model. The benefit is that quantitative relationships (mathematical operators) may be used where the underlying mechanisms of the model are known, and qualitative relationships (maps) may be used where there is only an expert belief and uncertain at that. The DSI method thereby accommodates subjective belief as well as mathematical operators in one system. Uncertainty of analysis as well as process variability are supported and propagated through the model. This chapter defines the usage of the terms qualitative and quantitative by extending the conventional definitions of scales of measurement, and proposes that there are conditions under which the combination of qualitative variables into a qualitative one may be valid. Both fuzzy theory and DSI are scrutinised for the validity of their conversion processes and it is concluded that DSI is the superior methodology. A mechanism is also described wherewith DSI is able to downgrade the information content in a variable.

6.1 Introduction

A characteristic of early design is the uncertainty of knowledge and the qualitative nature of data. Conventional engineering design tools are structured to apply mathematical relationships to quantitative variables, and therefore are difficult to apply to early design.

There is a need for tools that can process quantitative and qualitative variables, with their process variability through relationships that may be based on uncertain knowledge. This section describes how the Design for System Integrity (DSI) method permits both quantitative and qualitative variables in a model.

6.2 Classifying quantitative and qualitative

The terms quantitative and qualitative need to be set in context of knowledge and abstraction.

The knowledge issue is *uncertainty of analysis (incompleteness of knowledge),* which may be a mathematical expression at best, otherwise logical rules, or only (expert) opinion. Knowledge may therefore be either quantitative or qualitative. Knowledge may therefore be uncertain. Thus given certain inputs, there may be several outcomes predicted, with different probabilities of occurrence.

The abstraction issue refers to the nature of the variables, which may be either quantitative variables (ratio and interval scales) or qualitative variables (ordinal and nominal scales). These are described in further detail below. It is important to note that both types of variable may have process variability, i.e. the data may be uncertain. This is to be distinguished from uncertainty of analysis.

Quantitative and qualitative variables (i.e. abstraction of information) are now defined by an extension of the classification of Ackoff (1962), and shown in Figure 6.1. The change is an elaboration of the types of qualitative scale.

A ‘Quantitative’ variable is one on which arithmetic operations may validly be performed, which requires that it be a number on either an interval or ratio scale. All measurement points on a Ratio scale can be expressed as a ratio of other points, and the ratio scale can be infinitely divided (eg length) In contrast an Interval scale has a reference point (eg temperature 0 ^{o}C) from which all other points are expressed as a difference.

A ‘Qualitative’ variable is either ordinal or nominal in scale. An ordinal (or ordered) scale is one where the points can be ranked by comparison with each other (though not necessarily expressed as numbers), whereas a nominal variable is only a name to distinguish between objects without ranking them. Simultaneously, a qualitative variable is also expressed either as numbers or text.

This results in four combinations of qualitative variable: ordinal numerical (eg Moh hardness), ordinal textual (eg hot, warm, cold), nominal numerical (eg jersey number on sport players), and nominal textual (eg soil types of sauce, rice, jam). Thus an ordinal variable, even if expressed as a number (such as Moh hardness), is not considered quantitative.

To combine quantitative and qualitative variables in one model necessitates the ability to provide for various degrees of uncertainty of analysis in the model, since the qualitative variables cannot be processed by mathematical expressions.

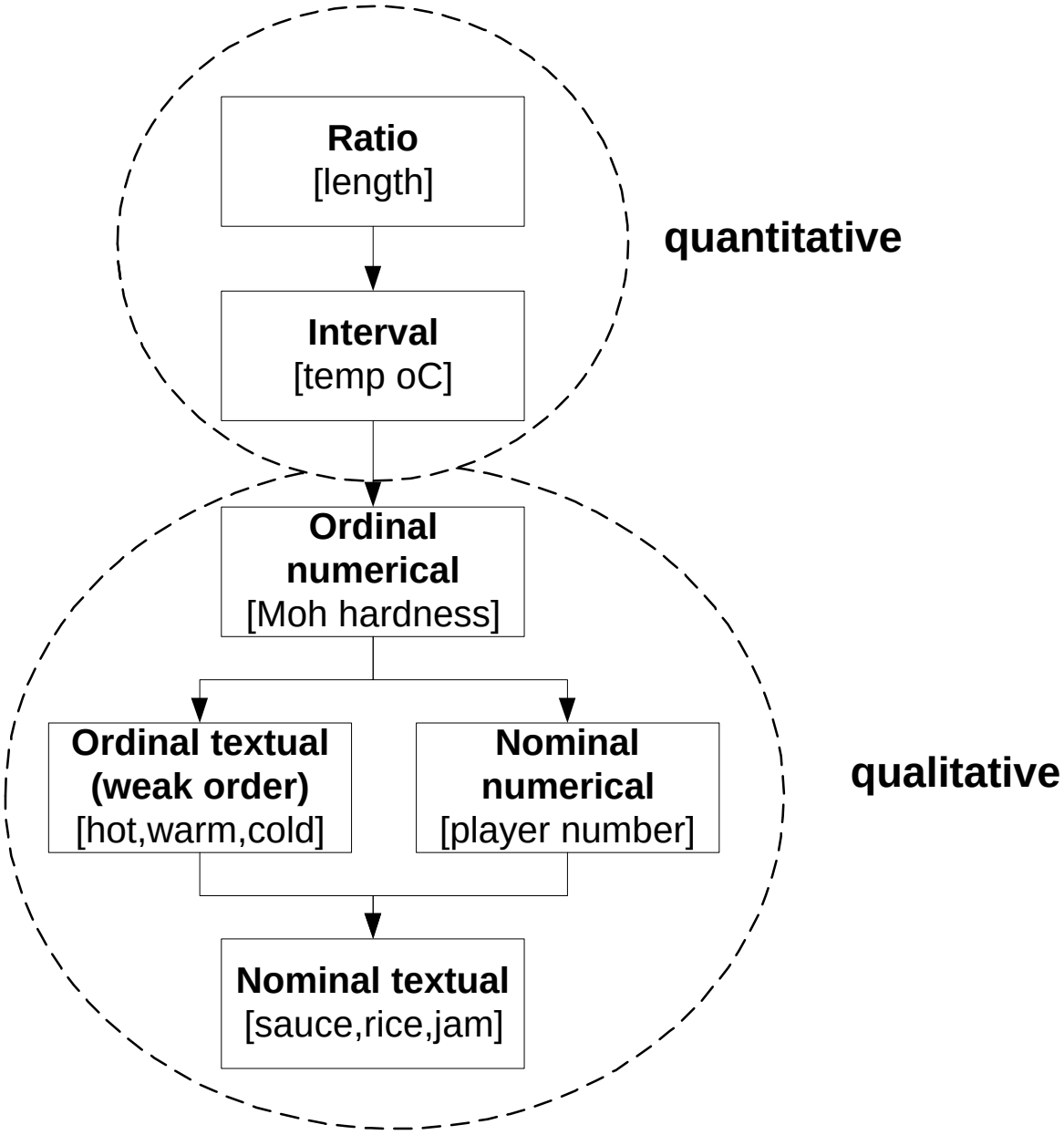


Figure 6.1: Classification system used in this thesis to distinguish between qualitative and quantitative variables. Quantitative variables present no special difficulty as they are those for which arithmetic operators are valid. The qualitative variables are ordinal or nominal, and simultaneously either numerical or textual. Examples of each type are given in brackets. The arrows indicate the direction of decreasing information content.

6.3 **Other approaches**

The ordinal textual scale is elsewhere called a weak order (Scott and Antonsson, 1999). A challenge in early design is to find ways to incorporate qualitative data, typically though not exclusively ordinal textual data, in modelling systems. Popular tools, such as QFD, fuzzy theory and genetic algorithms, convert an ordinal textual scale into a quantitative one, and then apply arithmetic operations to those numbers. Similar conversions occur in utility theory and multi-criteria decision making where multiple aspects of utility are aggregated into a single measure of worth.

The validity of these conversion process is questionable (Scott and Antonsson, 1999). How can one convert ‘hot, warm, cold’ to numbers in a defendable and consistent manner? Users of QFD, genetic algorithms and multi-criteria decision making seldom publish any defence, and it seems reasonable to conclude that the assignment of numbers to textual lists is subjective and possibly even arbitrary. Fuzzy theory is perhaps easier to defend, because it permits the user to define a range of uncertainty for each variable in the ordinal scale.

For example fuzzy theory would permit ‘hot’ to range from 45 to 100 with the most likely value being 65. In this way fuzzy theory acknowledges the uncertainty inherent in converting an ordinal scale to an interval one. The other systems do not provide this, as they require a single value (eg ‘hot’ = 65). Notwithstanding the lack of robustness in some qualitative to quantitative conversion processes, all the above methods are in widespread usage.

Yet more problematic is the application of these tools to nominal textual data, i.e. where there is no order at all and the variables are expressed as text. For example, how does one characterise the severity of soil given soil types of sauce, rice, jam, etc? This problem may arise when a global figure of merit or numerical order is required for multiple qualitative variables.

Is it impossible to convert ordinal to interval data, or for that matter any other conversion from a lower to a higher information content, against the arrows in the figure? In a knowledge vacuum it may indeed be true that qualitative information cannot be converted to quantitative data. However in many knowledge-based applications such as engineering design, there is a large body of knowledge available in the form of the designer's experience and knowledge coded into books, so the accumulation of multiple pieces of qualitative information may arguably make a quantitative prediction possible.[120] The critical factor is whether or not the process is based on background knowledge. If so the conversion needs to be defendable in a way that others can scrutinise and challenge.[121] If there is no background knowledge behind the conversion, then the assignment of numbers is presumably simply arbitrary and invalid.

It may be valid to convert qualitative data to quantitative providing that other knowledge is demonstrably available to augment the process. Methodologies such as QFD, AHP, genetic algorithms and multi-criteria decision making have briefly been discussed and it is submitted that because they produce a single point (crisp)

---

[120]If there is no other information than the textual scale being considered, then converting it to a higher information content, eg interval scale, would appear to be invalid. However if there are several sources of qualitative data then the aggregation of those data may permit a higher order information content. The following thought experiment explores this: is it possible to predict the ambient temperature for tomorrow given that a moderate North-West wind is forecast? The question calls for a number on an interval scale, and the information provided about the wind is purely qualitative. At first there would seem no easy answer. However, given the meteorology of Eath it is possible to give at least a generous range, eg. temperature could be -40 to +50 deg C. If other information becomes available, for example the season (summer) and location (Christchurch, New Zealand), so the range of uncertainty may be narrowed. Without ever living in that city it becomes possible to narrow the range to 5 to 40 deg C simply based on observing that the city is in temperate latitudes. Someone with meteorological knowledge or first hand experience of the climate would be able to further refine the range, eg 25 to 38 deg C. Thus:

(1) qualitative information (wind conditions, season & location) may be used to predict a number on an interval scale (temperature) by expressing it as a range of estimates, and

(2) as additional qualitative information is available so it becomes possible to contract that range of estimates.

When sufficient qualitative knowledge is available, then it is plausible that the interval could be contracted to deliver a single number of practical value.

[121]The concept proposed here is that conversion from a qualitative to a quantitative scale can be valid if there is other knowledge to assist the process, and where such knowledge is defendable. This concept is in agreement with the approach of Scott and Antonsson (1999) who state that there must be an explicit procedure for assigning numbers to weak orders.

conversion their users should be explicit about the assignment procedure. If this is lacking, as it unfortunately often is, then the assignment is presumably intuitive or even only arbitrary, and the results are likewise of limited validity. The probabilistic methodologies such as fuzzy theory and decision theory are more robust as they acknowledge the existence of uncertainty by providing a range of estimates with likelihoods attached. Even so their assignments should be defendable.

While *decision analysis* is able to process discrete qualitative values and their probabilities, and separately *Monte Carlo* is able to do the same for practically continuous quantitative values and their probabilities, the residual problem is that it is difficult to combine qualitative and quantitative variables in one model. Others have given this problem some attention: Wolstenholme (1999) examines the appropriateness of qualitative and quantitative modelling, and suggests that both methods need to be used in an intertwined fashion. If there is a single methodology that purports to achieve this it would be fuzzy theory. However fuzzy theory requires at least at ordinal textual data, and even then it can be hard to justify the assignment of numbers to the text. It is difficult to see how it can validly operate on nominal textual data, though there is nothing that prevents a user from doing so.

Berleant & Kuipers (1997) seek to combine qualitative and quantitative simulation, and used *semi-quantitative simulation* towards this. They used “numeric intervals to represent incomplete quantitative information”. Their system progressively refined a qualitative simulation, interpolating intermediate states, and producing quantitative outputs. Though they used the term qualitative, their methodology could only cope with ordinal numerical scales and not the other qualitative scales.

The DSI methodology provides some solutions to the problem of combining qualitative and quantitative probability simulation. This chapter describes how the methodology achieves this.

6.4 **Development strategy**

Decision analysis and Monte Carlo simulation are not immediately compatible. In principle a system could be devised that combined both analysis tools into one conglomerate, but this would only be for convenience and there would be no real interchange of data. Monte Carlo generates a random number between 0 and 1 (representing the cumulative probability) and then uses the inverse probability function, which is a mathematical expression, to determine the corresponding point on the quantitative scale. This value is then combined with one from another distribution using addition (or other operator) to determine one point in the output.

Monte Carlo cannot process qualitative distributions or qualitative relationships because its inverse probability function requires a quantitative mathematical expression and a quantitative scale to map to. Each single pass through the Monte Carlo algorithm is deterministic in that it generates only one output data point of unknown probability of occurrence, and the full probability distribution for the output is only apparent after many passes have been accumulated and placed into histogram bins.

To combine quantitative and qualitative simulation it was first found advantageous to develop an alternative methodology to Monte Carlo. Specifically, it was necessary to avoid the need for a quantitative inverse probability function. The solution was found in extending the decision table approach of decision analysis into the quantitative domain. The result is a system that takes two input histograms and combines them in combinatorial manner using the given mathematical operator. Importantly, the method does not need to know the mathematical expression for the input distribution, i.e. an inverse probability function is not necessary.

Another differentiating feature is that the method steps its way through a calculation, combining the whole of a few distributions at a time (usually two, sometimes three), whereas Monte Carlo determines only one final deterministic result in each pass. The convergence issues of Monte Carlo are therefore avoided. The development of the

separate quantitative and qualitative algorithms in DSI have been described in previous chapters.

6.5 **Creating quantitative data from qualitative**

The qualitative side of the DSI methodology is to use decision tables, which are here termed maps. Two qualitative probability distributions, each of which is a series of text descriptors (eg *Fresh, Dried, Reheated*) together with probabilities (eg 0.4, 0.5, 0.1 resp.), may then be processed through the map to determine the output distribution (eg *SolubleFilm, FineParticulates, LoosePieces*...), and its probabilities (eg 0.16, 0.2, 0.11....resp.). In this case the map would be converting two qualitative distributions into another qualitative one. This process is detailed in the previous chapter.

As discussed above, it appears reasonable to conclude that qualitative data may be converted to quantitative data where there is additional knowledge to support the process. DSI permits a map to be used as a *correlation* table. Thus it permits multiple statements to the effect of '*if inputs are A and B, then output is C'* to be condensed into a table. Correlation tables are a well established tool in other analysis systems such as Monte Carlo analysis, where they are used to correlate quantitative variables for which the usual assumption of independence does not hold. Indeed the DSI maps can be used for exactly this purpose too. However the DSI maps are more powerful than straight correlation tables as they:

(i) include uncertainty of the form '*if inputs are A and B, then output is range* C_i *with probabilities* p_i*'*, and

(ii) they permit the inputs to be either quantitative or qualitative.

Thus the maps in DSI combine the functionality of both quantitative correlation tables and qualitative decision tables. They could be considered an extension of correlation tables towards qualitative variables, and equally validly as an extension of decision tables towards quantitative variables.

An example could be useful to explain these features, and for this we consider a fragment of a wash performance model. The full model is discussed in a later chapter, but for now we focus on the fragment shown in Figure 6.2.

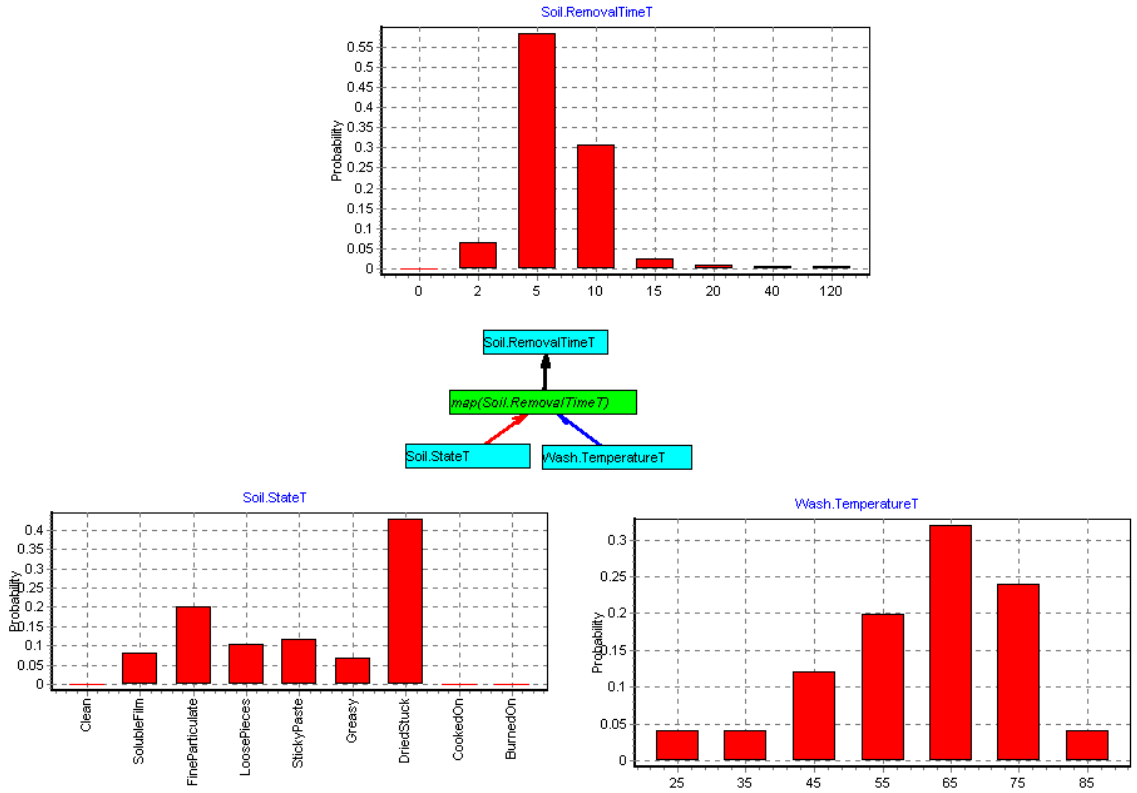


Figure 6.2: Model used to determine soil removal time (A) using a map. The graph in the centre of the figure shows the operation. The inputs are the two distributions at the bottom of the figure, and the output is the top distribution.

The input called *SoilStateT* is a variable describing the condition and type of soil on dishware that are placed in a dishwasher. It is entirely qualitative, and to use the terminology established above, it is nominal textual since there is no ranking of the elements. The other input is *Wash TemperatureT* and is quantitative (interval scale) or qualitative (nominal scale).

The map correlates these two inputs according to other knowledge, and uncertainty of that knowledge, that the expert holds and has expressed in the map. A fragment of the map is shown in Figure 6.3. An example entry is soil state ‘clean’ and wash temperature ‘25' results in soil removal time of ‘0'. In this case the entire probability of one is assigned to output ‘0', which reflects an expert who is certain of belief. However in general the probability could be distributed across multiple outputs if there was uncertainty of analysis, and this is shown in other columns. For example with soil state ‘Sticky paste’ and wash temperature ‘25' results in soil removal times that range from 5 to 20, with probabilities assigned to each.

Soil.Remo			Soil.State(								
			Clean	SolubleFilr	FineParticl	LoosePiec	StickyPas	Greasy	DriedStuck	CookedOn	BurnedOn
Wash.Tem	25	Comment									
		0	1	0	0	0	0	0	0	0	0
		2	0	0	0.2	0.2	0	0	0	0	0
		5	0	0.25	0.2	0.4	0.2	0	0	0	0
		10	0	0.25	0.4	0.4	0.2	0	0	0	0
		15	0	0.5	0.2	0	0.4	0.25	0.25	0	0
		20	0	0	0	0	0.2	0.25	0.25	0	0
		40	0	0	0	0	0	0.25	0.25	0.5	0.5
		120	0	0	0	0	0	0.25	0.25	0.5	0.5

Figure 6.3: Fragment of Map used to correlate two input variables, soil state and wash temperature, to soil removal time.

The map makes no assumptions as to the type of input or output variable, whether nominal/ordinal/interval or textual/numerical. It simply treats all inputs as text strings, and applies the knowledge contained in the table to determine the output distribution. In this example the output is *Soil removal time T* which is a series of numbers which a human user would interpret as an interval scale and therefore quantitative. However the map itself made no such interpretation. Instead it simply treated all inputs and outputs as text strings. If there is a higher order interpretation in those strings, as here, then that is a consequence of the knowledge the expert has coded into the map. In this way a map provides a mechanism for an expert to express opinion and knowledge (and uncertainty thereof), and even to use qualitative data to come to an uncertain quantitative conclusion.

It is acknowledged that a map could be used inappropriately by the user, i.e. to produce a quantitative output based on arbitrary and invalid assignments. The DSI method would have no way of knowing that was happening, nor could it preventing such abuse. However this limitation is common to all engineering analysis tools, (including Fuzzy theory), as they too rely on the professional judgement of the user.

In the map example shown here, a mixture of qualitative and quantitative inputs have been used to produce a quantitative output. Other combinations of inputs are possible. Two of these cases have special names: two qualitative inputs producing a qualitative output is a decision table, and two quantitative inputs producing a quantitative output is a correlation table. The map methodology of DSI accommodates these two special cases as well as other ways of combining qualitative and quantitative data.

Some might reason that a similar process is available through fuzzy theory, so the differences will now be discussed. The primary difference, and it is a significant one, is that fuzzy theory does not provide decision tables or correlation tables. Fuzzy theory provides an approximate probabilistic computation mechanism, and it can operate on qualitative data in a limited manner. It transforms a qualitative input to the quantitative domain, applies an arithmetic operator to produce a quantitative output, and then transforms that back to the qualitative domain. DSI solves the problem entirely in the qualitative domain, and is therefore not constrained by the need to apply transformations.

A more detailed description of the differences follows:

- For fuzzy theory to combine two qualitative probabilistic variables[122] it first has to convert each one to a number on an interval scale. This constrains fuzzy theory to ordinal (numerical and textual) scales. It cannot easily be justified to operate with any form of nominal data since these do not convert to interval

[122]The term ‘distribution’ might be natural to use here, but fuzzy theory avoids that term, instead preferring ‘membership function’, and indeed they are not quite the same thing.

scales. In comparison the DSI maps can operate on all forms of data including nominal.

- Once the inputs to a fuzzy model have been converted to an interval scale, then the fuzzy arithmetic can operate. Only mathematical operators on numbers are supported. DSI supports mathematical operators, and decision tables too.
- The output from a fuzzy calculation is a set of numbers again on an interval scale, and if necessary this can be translated to an ordinal scale, but not to a nominal scale. DSI not only produces a qualitative output directly without such translation, but it can produce any kind of qualitative output including nominal.
- The probabilistic computation used in fuzzy theory is not consistent with the algebra of random variables nor Monte Carlo analysis, whereas DSI is.

On the basis of the above features, DSI provides a more comprehensive mechanism to combine qualitative and quantitative variables than is available in Fuzzy theory.

The author does NOT suggest that DSI be used to a make a one-to-one conversion of a qualitative variable into a quantitative variable. Such a feature is an essential part of processes such as QFD and multi-criteria decision making for example, and there are constraints on the validity of the approach. The author suggests that a more appropriate and robust method is to use the map method of DSI to combine multiple sources of qualitative information into a single output variable based on defendable knowledge.

The map approach of DSI provides the following benefits:

(1) The map provides a traceable and defendable record of the expert's opinion.

(2) The existence of uncertainty of analysis is acknowledged and formally represented in the map.

(3) The map accommodates all forms of quantitative and qualitative variables, (providing of course that the user has valid grounds for including those variables).

There are practical considerations to using the DSI software, and as these are only mechanistic they are provided in Appendix 1. One-dimensional maps are also permitted, and these may be used as look-up tables or for *correlation* of variables (to be discussed in a following chapter).

6.6 **Degrading the information content**

The conversion of a higher order scale to a lower one, eg an interval scale (1,2,3) to a qualitative one (eg 'good,ok,bad') is not a contentious one as the information content is simply being degraded. One might question why an analyst would want to degrade the information content in a model. The process is necessary when the distribution contains too much information to be useful or practical to process. Sometimes the person using the simulation results wants to have a simple textual output regardless of the internal detail of the model.

DSI maps provide a way in which information content may be downgraded, eg the conversion of a interval scale to an ordinal textual one may be done this way. For example, if a quantitative part of the model calculates mass in kilograms, then a map can be used to take that input and produce a textual description such as 'Light, Medium, Heavy', complete with uncertainty. The uncertainty in the mass in kilograms (expressed as a quantitative probability distribution) will be converted by the map into a qualitative probability distribution.[123]

It is therefore possible to take a quantitative result and downgrade it to a qualitative distribution relatively easily. Figure 6.4 illustrates the process whereby a quantitative

[123]Assume that mass is calculated using an arithmetic operation. The mass will then be a ratio scale and will have a probability distribution with perhaps 100 data points in it. This distribution is then input to a map, one axis of which has bins for mass labelled '5, 20, 50, 100'. The number of bins need not correspond to the number of incoming data points. The Integrity-T algorithm has been provided with sufficient sense to detect an incoming numerical distribution and re-bin it into the map groups. For example the probability file '1, 3, 5, 7... 99' would have its probabilities redistributed to bins correspond to the '5, 20, 50, 100' of the map. No user intervention or preconditioning is required for the conversion since the numerical file already has an origin.

variable (*Dishware.Daylight*) may be input to a map to determine an output (*Wash.DirectFraction*).

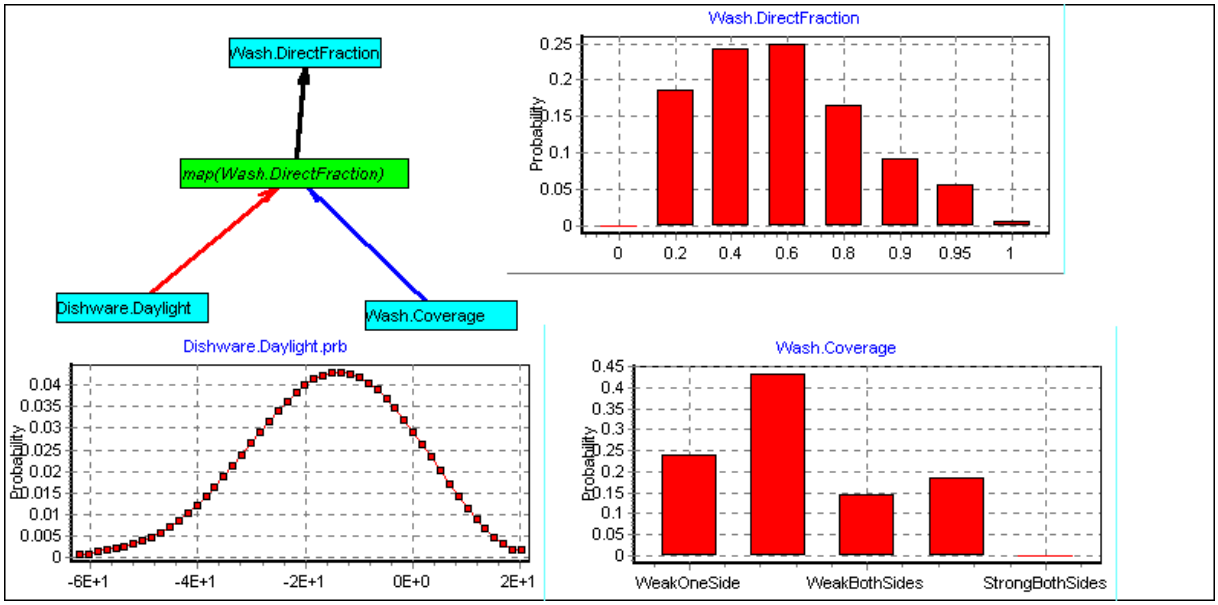


Figure 6.4: Bridging from quantitative to qualitative variables. The graph model (upper left) uses a map to combine two nominally qualitative variables. In this case one of the variables (Dishware.Daylight) at lower left is quantitative since it is the result of a previous computation (not shown). The map expects only a few values of Dishware.Daylight, not as many as present in the quantitative input. However the software copes with this by distributing the quantitative input into the coarser quantitative bins and then proceeding with the map computation. The other input (lower right) is purely qualitative. The final output at top right is qualitative (viz the histogram bars) but the map has given its text labels a numerical interpretation so it is eligible to be input to subsequent quantitative operations.

6.7 **Conclusions**

The value of the DSI methodology is that quantitative relationships (mathematical operators) may be used where a model is well enough understood, and qualitative relationships (maps) may be used where there is only an expert belief, and uncertain at that. The information content in a variable may readily be downgraded (eg an interval scale to an ordinal one) using maps. This is a valid conversion, and it may be

necessary where the primary variable contains too much information to be useful or practical.

Upgrading the information content of a variable (eg an ordinal variable to an interval one) has constraints operating. Some might say that such a conversion is impossible, and to a large extent the author agrees with that view. However it has also been shown how multiple sources of qualitative information can validly be used to make quantitative estimates. Therefore it is submitted that the DSI map process, if used correctly by the user, can result in a valid and defendable combination of qualitative variables into a quantitative one.

It is not necessary to create one model for the qualitative aspects and a separate one for the quantitative. Qualitative and quantitative relationships and data may be mixed through even one model. Consequently systems may be modelled and simulated, even if knowledge is incomplete or held with different degrees of conviction. The case of wash performance of dishwashers illustrates this, and is further developed in a following chapter. There are large uncertainties of analysis (underlying principles are unknown) as well as process variability, but providing an expert opinion can be found, then these uncertainties may be represented and modelled.

These features enable the DSI methodology to be used wherever there is uncertainty. Following chapters demonstrate the capabilities of the method in the dishwasher domain, and particularly show how it can be used at early design where information is sparse and even uncertain. A tightly integrated methodology provides the capability for this. The methodology is embodied in software which makes it easier for the non-statistician to engage with the probability concepts, and eliminates the computational drudgery that would otherwise exist with a purely paper-based implementation.

Chapter 7

Multiple viewpoints of design

This chapter describes the application of a multi-viewpoint approach to design, whereby different aspects of the design may be simultaneously and semi-automatically modelled in parallel with the primary functional mode.

7.1 **Introduction**

Research and development (R&D) is critical for the sustained business profitability of engineering organisations. However the R&D activities are difficult to *manage*, as the nature of the work is inherently open-ended, and carries large uncertainties and risk. Typically there are technical as well as financial risks.

An essential task of anyone who designs or manages design is to ensure the integrity of the product and its function. Integrity is measured in *key characteristics* of function, safety, cost, and reliability among others. Some key characteristics are determinable during the design stages of product development, while others only become apparent in the finished product. The task is to ensure that these are identified and have a high enough likelihood of being positive. There is advantage in being able to predict an outcome (deterministic), more in predicting the probability of that outcome, and the most advantage in predicting the distribution for the outcome.

The ability to anticipate likelihood of outcomes is important in understanding the risks and identifying how robust the input estimates might be. The sooner in the design process this can be done the greater the opportunity to change the outcomes. As the key characteristics span multiple viewpoints, so it is necessary to be able to scrutinise those other viewpoints.[124]

Integrity of the key characteristics is important as it affects the customer's perception of product worth, and therefore sales and corporate success. Thus there is a need to design for key characteristics as perceived by the customer, including the ability to predict key characteristics in multiple viewpoints early in the design process. Failure to do so may result in product failure, and Martino (1994) identified two ways in which this might occur: 'the risk of early *obsolescence* of the product' brought about by setting the performance goals of the product too low thereby giving room for a better product from a competitor, and 'risk of project failure' caused by performance goals that are too high and which encourage technically risky solutions.

[124]The concept of life cycle issues is also encompassed here.

Engineering design is a complex task because key characteristics exist in various viewpoints, and have different measures of worth. It is not only the functional requirements[125] that need to be satisfied (eg. a dishwasher must wash dishes adequately), but also other aspects including reliability, manufacturability, and cost to name a few. For example Rosen et al (1994) discuss "functionality" as a primary viewpoint, with the need to evaluate designs in the secondary viewpoints of "manufacturing, cost, and other life-cycle considerations". A machine that has intrinsic design *integrity* is one that robustly provides the required or reasonably anticipatable functions in multiple viewpoints despite uncertainties in the designer's understanding of the mechanisms, production process variability and the uncertainties of user environment. As Wallace (1987) noted, the criteria for successful design are balance in the execution of the design process, paying attention to the effect of key factors, and integrating different types of knowledge.

High design integrity reduces the downstream life cycle costs for all stakeholders in the product. However it is often difficult to anticipate the multiple viewpoints while in the early design stages. Integrity is partly if not primarily determined at concept design. It is at this stage that many choices are made, that shape the final solution. The constraint is usually in terms of geometry, which has downstream consequences on manufacturing process, failure modes etc. It is increasingly difficult to change the large-scale geometry as the design progresses.

Despite a general awareness of the need to consider the life cycle costs (eg Finger et al, 1992)[126], the methodology for estimating key characteristics and their uncertainty

---

[125]The primary viewpoint is usually the designer's concern with function, that the product should provide the functional performance required by the specification. This is a necessary viewpoint for the success of the product, since it must fulfill the needs for which it is to be created. However the exclusive optimisation of a limited number of functional attributes can too easily occur in design. Other attributes need to be considered early on in the design process, such as manufacturability, reliability, safety, and indeed potentially many other life cycle issues. There is a risk of the designer overlooking effects in viewpoints other that those being worked on. A design change that boosts performance of the machine (eg raising operating temperature of a dishwasher), may have adverse consequences in the reliability viewpoint.

[126]Finger et al (1992) proposed a software solution aimed to create a "computer-based design system that will enable a designer to consider concurrently the interactions and tradeoffs among different, even conflicting, requirements". Their proposal was to use software "experts and

from multiple viewpoints is poorly developed, especially for early design, where it could potentially be of greatest value.

The provision of risk data from multiple viewpoints is valuable for ensuring that decisions have integrity. This chapter explains how the Design for System Integrity (DSI) methodology achieves this.

7.2 **Creating additional views as a by-product of functional modelling**

The intention of this project was to promote design *integrity* across multiple viewpoints, by providing means to help designers understand the interactions between devices in their system.

Each viewpoint is a *functional model* represented by a block diagram or graph. The integrity of the machine is determined by the outcomes of these multiple viewpoints. For example in designing a dishwasher there could be viewpoints for each of cost, wash performance, reliability, safety and noise. All these views would be active at once, sharing data as necessary. In contrast to some of the other proposals such as Finger et al (1992) for systems with sufficient artificial intelligence to be able to critique a design as it develops, the current work aims simply to provide the framework on which the designer can develop the functional model. The author has taken this less interventionist approach as there is little evidence in the literature to show that the artificial intelligence tools have had significant usefulness in early design. The more successful tools appear to be those that do not attempt to direct the designer.

The multiple view capability was implemented in the DSI software. Previous chapters have described the development of the DSI methodology and its probabilistic computation algorithms. What might not have been explicit is that the software allows

---

advisors, called perspectives" to comment on the design as it progressed. The proposal does not yet appear to have progressed to a working system.

multiple viewpoints (eg cost and reliability) to be open at once. The semi-automatic generation of other viewpoints relies on this multi-view capability.

The DSI software was extended so that placing a representation of a physical device (eg ‘WashPump’) in any view, automatically creates an entry in other views (providing a catalogue is selected beforehand). Also, each additional entry is automatically assigned default data according to the domain the designer is working in. Consequently, while the designer is busy creating for example a probabilistic model wash performance for the product, the software creates elements of the cost, reliability, safety and noise (etc.) models in the background. When the designer transfers his attention to one of these other views, the representations of the devices are already in place, see Figure 7.1, with default data behind them. The designer may then work that view into a suitable model. The designer has to put in the logical structure, that is the quantitative and qualitative relationships.

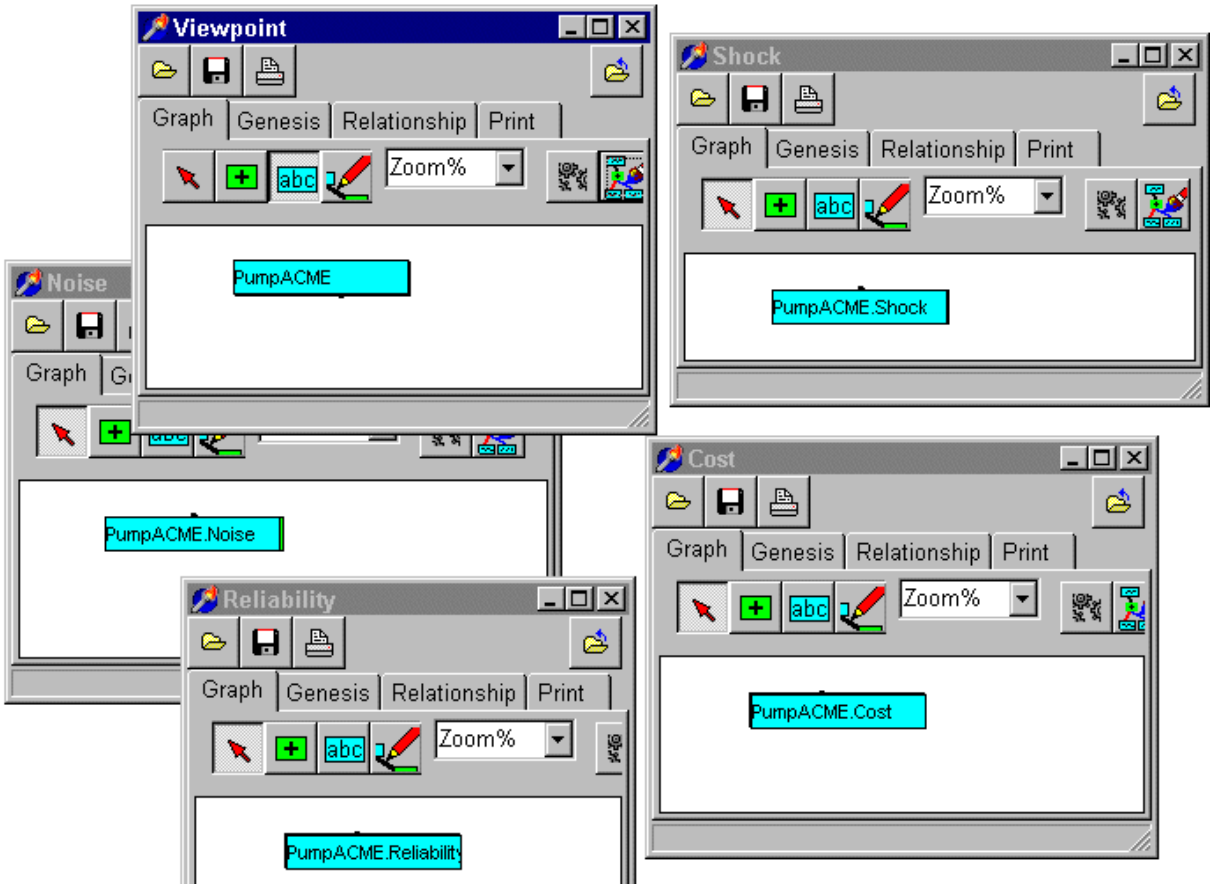


Figure 7.1: Multiple views created by the DSI are shown here. The original view was 'Viewpoint' and a device 'PumpACME' was placed on it. Since the type of the device was set to 'WashPump' which has data for multiple views in the domain file, the system automatically created all the other views (windows) and placed the device there too. There is default probability data behind each device.

The default data are extracted from a domain specific data file. This catalogue contains a list of all the devices with their views and their data in that view. For example 'WashPump' exists in the dishwasher domain, where it has several entries ('WashPump.Cost', 'WashPump.Reliability', 'WashPump.Shock', 'WashPump.Noise') each with data. The user can add more data to the catalogue, for example a new type of pump called 'PumpBrandACME' with all its characteristics.[127]

[127]The catalogue is termed a domain file, given a 'dom' extension, and may be edited with any text editor.

When the user places a box (eg ‘PumpACME’) in any view of the model, then the system prompts for the type of device (eg ‘WashPump’) and then retrieves all the data for that type from the catalogue, see Figure 7.2. Since ‘WashPump’ has been defined to have cost, reliability, shock and noise views, those views are created if they are not already open, and a block called ‘PumpACME.Cost’ (etc.) is placed in each view with its data. The designer can then edit that data as necessary. By providing default data the system attempts to assist the designer who is at an early stage and does not yet have accurate data or may not know what values would be reasonable.

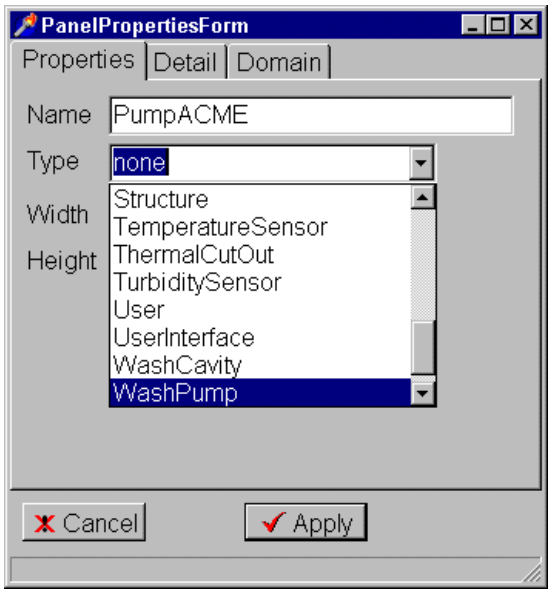


Figure 7.2: On placing a device (eg ‘PumpACME’) on a view, the system prompts for the type. In this case the user selects ‘WashPump’ and will therefore get all the views and data that belong to that generic type.

In the catalogue it is possible to have data for multiple candidate devices. For example there could be data for several different brands of wash pump. The multi-view capability in DSI allows the type of an existing device to be changed, eg from one brand to another. In doing so the software updates the data in all the other views (cost, reliability, etc.) using the new type. New viewpoints may be added to the catalogue.

In chapters 10 to 16 various multiple viewpoints of dishwasher performance are examined individually.

7.3 **Discussion on multiple views**

Catalogue design system

The existing DSI software has been connected to a catalogue of standard devices, eg pumps, hoses, seals, specific to the domain of study. For each device the parameters are given for several standard viewpoints such as cost, reliability, etc. It has been arranged that selection of one of these devices causes the parameters to be updated wherever they appeared in the model. This permits the easy substitution of one device for another, without having to manually edit device properties. The benefits are convenience and the ability to use a standard *catalogue* of devices.

The catalogue contains domain specific knowledge, including the types of other viewpoints, and the default probabilistic parameters for the variables in those viewpoints. It is important to note that the catalogue is not coded into the software but is a separate file. Thus the DSI multiple view approach may be applied to any domain for which a catalogue may be created.

This begs the questions as to how a catalogue file would be created. Two methods are suggested. The first method would be to use a past DSI model (with or without editing) as the catalogue for the next generation of development.[128] In this way design knowledge of similar previous machines may be re-used, thus supporting the accumulation of organisational learning. By re-using past domain files in a boot-strapping process, it is proposed that an organisation might increase its knowledge of probabilistic properties for the components that it commonly uses.

But how would an organisation begin, lacking probabilistic performance data? This is not an easy question, and it may be one of the reasons why, as Thornton et al (2000) discovered, United States manufacturing industry typically applies risk

[128]Technical note for software utilisation: To use a past DSI model as catalogue, the earlier design needs to have been from multiple viewpoint which have been combined into one file. An appropriate menu selection on the DSI software will automate this process. The resulting ‘int’ file may then be opened as a domain file when devices are placed in a new project. Otherwise edit the prior ‘int’ file with a text editor, remove superfluous data, and save as ‘dom’ file in plain text format.

management poorly, late, or not at all. There is a need to create corporate repositories of probabilistic data on component performance, process capability and other *key characteristics* that affect the end product. As an awareness of risk grows, it is likely that component vendors must increasingly provide original equipment manufacturers (OEMs) with reliability and probabilistic performance data for their components. In time this might be sufficient to stock a catalogue like that of DSI, but at present there is insufficient information commercially available to make this an easy task.

However the DSI methodology is comfortable even if there is no commercially available probabilistic data. An expert could list the standard devices (eg pump), name their viewpoints (eg cost, power, pressure, reliability), and provide default probabilistic parameters in each of those viewpoints. It would be prudent for the design manager to ensure that initial estimates had large dispersion in the probability distributions, so that the uncertainty was conservatively modelled. The parameters could then be narrowed as evidence was accumulated. This evidence would arise from measurement of process variability, reliability and other performance tests. The DSI software relies on external agents to provide the statistical manipulation or reliability analysis necessary to massage test and production results into distribution parameters that can be utilised in the software. However the software can accept histograms since its file format is open and can easily be mimicked by a spreadsheet or wordprocessor.

'Signposting'

The design manager would presumably like to determine which parameters would, if tested and the uncertainty minimised, give the best consequences to the overall product performance. This need is recognised in the *signposting* methodology of Clarkson and Hamilton (2000), and in sensitivity analysis and the backward analysis capability of fuzzy theory. However it seems that full probabilistic computation methods, including the algebra of random variables, Monte Carlo analysis and DSI, cannot provide this.

Nonetheless the DSI method can predict the outcome of an improvement in design knowledge, and may be used to explore the design space for this type of information. Once better (narrower range) information is available for a parameter, then the software model may be recomputed. If appropriate the catalogue may also be changed, so that the information becomes persistent and available to future product developments. [129]

Automatic viewpoint creation

An extension or parallel development of the catalogue concept has been a degree of automatic viewpoint creation when a device was selected from a catalogue. For example, the designer can draw up the functional flow model using the DSI software, eg a fluid circuit, by selecting a pump and other relevant devices from the catalogue. The act of placing a device like a pump in any viewpoint automatically creates other viewpoints in the background, say for reliability and cost, with default pump attributes appearing in those viewpoints. It is intended that this would assist the designer to encompass in his thought processes those other viewpoints that otherwise might be neglected. The device is placed in the viewpoint for the designer to connect up with the appropriate relationships. The benefits are support for the semi-automatic anticipation of constraints.

Probabilistic computation

DSI also includes quantitative and qualitative probabilistic computation algorithms whereby uncertainty can be propagated through a model. The model in each view has access to these same algorithms, so it is possible to concurrently develop models for wash performance, cost, reliability, etc., using mixtures of quantitative and qualitative models. The presence of other views and the links between them are more explicit than with other functional modelling systems. Any data shared by multiple views is immediately available to other views when changed.

---

[129] Future developments with DSI might be able to explore the feasibility of doing a check in the background to find those parameters that would be most beneficial in reducing the overall uncertainty. This problem is reminiscent of both multi-parameter sensitivity analysis and optimisation, but is potentially more difficult than either as variables are probabilistic.

Using multiple viewpoints to help manage design decisions

To manage is to make decisions. However one of the difficulties in design decisions is anticipating the other viewpoints. Even with the best intent it may be difficult to do this, especially as decision makers may be focussed on the project rather than the bigger picture with its peripheral views. Also troublesome is that decisions often involve qualitative parameters as well as quantitative[130] ones (usually financial value), and it may be difficult or even impossible to put these onto the same scale for comparative purposes. In such cases it is difficult to balance the qualitative and quantitative considerations. [131] Another difficulty for decision making in general is the attitude of the decision maker towards risk. People are said to be *risk averse* (Clemen, 1996) if they select the option with the least risk even if the expected value[132] is lower than other choices. These conservative design decisions potentially result in a product that is solid but boring, uses more material, is heavier and larger, costs more, and has difficulties in a competitive market. There are others who are *risk phobic*. They are uncomfortable with any form of uncertainty, and treat all the options as deterministic.[133] The risk phobic approach is incompatible with creative

---

[130]The interested reader is referred to Chapter 6 for a description of how the terms quantitative and qualitative are used in this work.

[131]Even with the best financial information the consequences of the decision can still go the wrong way for the design team if other viewpoints are not considered. For example in the Malibu motor car (General Motors) the fuel tank was positioned too close to the rear bumper, increasing the risk of fire in a rear-end collision. A product liability award of US$9.4 billion (the highest ever) was awarded to six people who suffered explosion of their vehicle. 'GM had numerous failures in their crash tests, but chose to leave the tank where it was, because changing it would have cost $8.59' compared to the expected cost of $2.40 per car to settle liability claims. (Otago Daily Times, 12 July 1999, p8). If anything GM might have been too preoccupied with the financial viewpoint when deciding where to place the fuel tank, since it is clear from product liability law that liability requires (among other factors) that the manufacturer knew of the defect and could have done something to reduce the risk. Perhaps GM failed to consider the qualitative aspects of the legal viewpoint when making the decision?

[132]Expected monetary value (EMV) is often used in decision analysis, eg Clemen (1996). The EMV of a decision is the sum of the product of monetary outcomes and their respective probabilities. However EMV requires both monetary outcomes, and quantitative probabilities, and these may be troublesome to establish.

[133]They might apply a filter process to the decision, admitting as candidates only those solutions that are close to being deterministic, and suppressing solutions that have large uncertainties. Risk phobic behaviour may involve the decision being set up as a choice-of-one, i.e. seeking endorsement for a particular decision without presenting alternatives at all, perhaps claiming that none of the alternatives are feasible. Possibly this approach exists because of a individual's controlling personality or a culture of conformity. This is comparable to the 'most-probable-future criterion' of Blanchard and Fabrycky (1998, p 159). An example of significant failure might be the Boeing 747

design (cf brainstorming), and can cause catastrophic product failures. A *risk-seeking* approach applies if a person selects options that have high benefits, even if the expected value of those benefits is lower than other candidates (Clemen, 1996). A *risk-ignorant* approach applies if the decision maker fails to ensure that likelihoods are included in analysis, or if too few viewpoints are considered. It could simply be that the decision maker is reluctant to commit to quantitative estimates of probability, perhaps because there are no meaningful data. (There are several alternative approaches if probabilities cannot be assigned.[134])

The integrity of a decision may be enhanced if the decision makers are able to objectively consider their attitudes to risk, and move towards greater rationality if appropriate. The ideal rational decision maker is a person who is risk-neutral (which implies that risk is neither a deterrent nor an attraction in itself), has considered all options regardless of their state of development or uncertainty, and has a robust sense of purpose (possibly a company mission statement) that permits qualitative parameters to be considered in the decision alongside quantitative ones, even if the two cannot be directly compared.[135] In addition the ideal decision maker has actively sought information about the consequences (and their likelihoods) in multiple viewpoints, not only the primary viewpoint.

---

aircraft design that positions the centre fuel tank too close to the hot air-conditioning unit. This fault appears to have been the cause of the loss of flight TWA 800 in 1996. However Boeing knew of the fault since 1980, but failed to take corrective action due to problems with its internal culture (Ignatius, 1999). Likewise the inability of NASA management to respond to known O-ring problems caused the loss of the Challenger space shuttle in 1986.

[134]Available methods include *Laplace* criterion (treat all outcomes as equally likely), *maximin* criterion (for each scenario find the minimum or worst case result, and then select the scenario where this is maximum), maximax criterion (for each scenario find the most optimistic result and then select the scenario where this is highest), minimax criterion (determines the regret as the difference between each result and the best result, then determine the maximum regret for each scenario and select the scenario where this is minimum), and Hurwicz criterion (select a degree of optimism with which to weigh the best and worst results for each scenario) (Blanchard and Fabrycky, 1998, p 160-164; Taylor, 1999, p541-544).

[135]This thesis makes no attempt to convert multiple figures of worth to a common quantitative scale. The author believes that conversions such as qualitative worth to financial value can be difficult if not impossible to defend if they are included in overall worth. However excluding qualitative viewpoints from a decision, simply because they are not quantifiable, is to ignore important information. Instead it is necessary to apply the faculties of human judgement, and in turn this requires some form of higher conscience or mission.

DSI cannot actively enforce a risk-neutral approach to design decisions, but it does provide a mechanism to model risk and subjective beliefs, and this is expected to be valuable in itself. DSI actively supports modelling from multiple viewpoints, and thereby assists the design process.

7.4 **Conclusions**

The DSI methodology and software provides explicit support for multiple viewpoints, by partially automating the creation of viewpoints other than that which the designer might be working in. It achieves this by referring to a catalogue file, which contains information necessary to create and populate the other viewpoints.

As the design progresses and the designer can provide more supporting data, eg on actual failure rates, process variability etc., so then the results of a probabilistic computation would be expected to show increasingly narrow distributions. The extent to which distributions are wide and have been modelled at all provide a means to assess and manage the uncertainties in the design.

Chapter 8

Design for System Integrity Software

The operation of the Design for System Integrity software is described here from the user's perspective. Setting up a probabilistic computation model is described.

8.1 Introduction

The Design for System Integrity (DSI) software is a simulation tool that may be used to analyse a variety of problems. It was developed to model various parameters that are of interest to engineering designers, such as cost, machine performance, and reliability. However the software and the methodology are applicable to other problems. This chapter describes the operation of the software, from a user's perspective. For additional details about the software please refer to Appendix 1.

8.2 Representing the analysis problem in software

DSI requires that the user explicitly define each stage of calculation, and what the input distributions are. This is done using the various tab sheets on the user interface.

8.2.1 Creating a graph of the problem

The first step for the user is to create a graph of the problem by using the visual tools on the 'Graph' tab sheet. Figure 8.1 shows what this tab sheet looks like.

For example the expression *Profit = Income - Cost* is represented by placing three 'device' blocks and naming them '*Profit', 'Income',* and *'Cost'*. To do this, select the 'abc' button and then click anywhere on the canvas to place a device. Change the name to '*Profit*' in the small form that pops up. Repeat to place *'Income',* and *'Cost'* devices.

Device blocks must have unique names as each block will be a probability file. The characters in the name must also be acceptable as file name characters under Windows. To delete a block, right click on it and select the delete menu item. To move a block select the arrow button and then drag the block with the mouse.

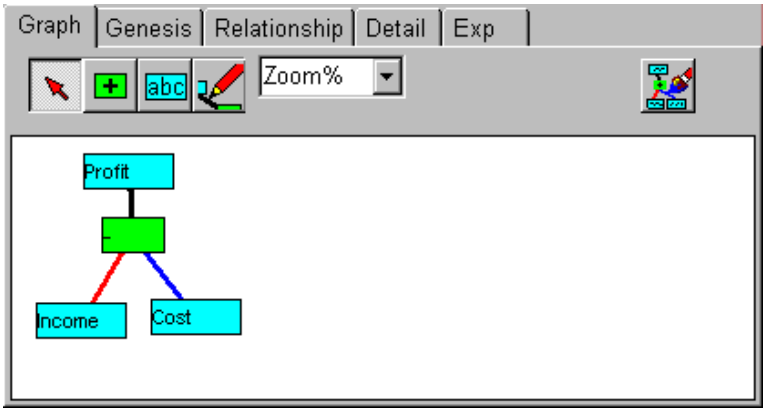


Figure 8.1: The Graph tab sheet is the primary interface for the user. On its canvas the user draws the graph representing the problem. The graph here shows that Profit = Income - Cost. This is created using the buttons on the toolbar.

Next place a 'relationship' block and name it '-' (abbreviation for 'minus'). To do this select the '+' button and then click anywhere on the canvas to place the relationship. Change the name to '-' in the small form that pops up. DSI recognises several types of mathematical operators, and often synonyms for each, as described in Appendix 1.

Then connect the blocks together with arcs. To do this, select the arc button and then click on the start and end blocks. Starting with the top level block ('*Profit*') is simplest as the first arc drawn will be identified as the output part of the group. This may be changed later if necessary. Then draw the arc between *'Income'* and the operator, and select the priority from the checkbox that appears. Generally each computation group in the model requires two inputs, one operator, and one output. The sequence of operators is black for output, and then red-blue-green priority for the inputs. In this example the inputs must be sequenced *'Income'* (red) and then *'Cost'* (blue) to get the minus operator correct. In cases such as plus and product the inputs can be in any order. Figure 8.1 shows the results at this stage.

Save the file by pressing the save button on the top level tool bar. It is recommended that the name of the top block, eg '*Profit*' be used[136].

8.2.2 Defining the input probability distributions

Once the graph has been defined, the next stage is to define the probability distributions for each input. Change to the 'Genesis' tab sheet where these will be made. The inputs (*'Income'* and *'Cost'*) are defined now in terms of say the Normal distribution. To do this type 'normal' into the Genesis Type column, and then the mean and standard deviation in the following columns. The software also needs to know how far out the distribution is to be modelled, and therefore requires the number of standard deviations (on each side) in the following column. A value of three is usually appropriate. Set the *'Income'* to normal, 1000, 200, 3 and *'Cost'* to normal, 600, 150, 3. The results should appear as in Figure 8.2.

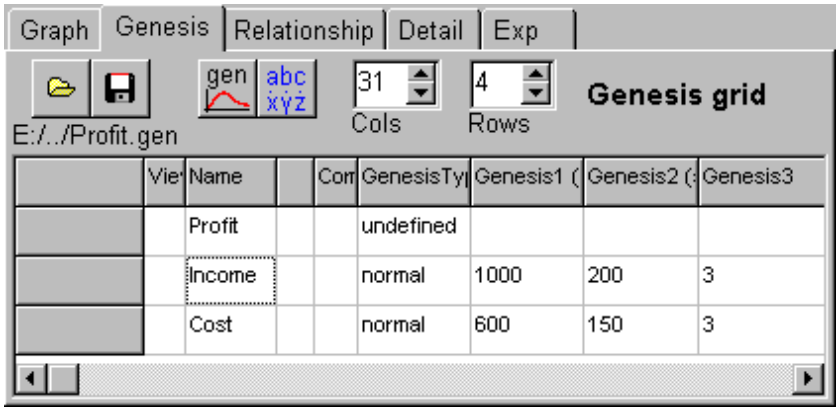


Figure 8.2: The Genesis tab sheet is used to define the inputs in terms of probability distributions. For example the grid here shows that Income is a Normal distribution with mean 1000, standard deviation 30 and that it will be modelled as far out as 3 sigmas (standard deviations) on each side of the mean.

Having defined the distributions they next need to be generated. Press the 'gen' button to generate all the distributions listed. Alternatively a mouse click on each cell labelled 'normal' will work. The distributions are automatically saved to disk using the

[136]The version 3 save process will save three files to disk (*.int, *.gen and *.rel) , and it is recommended that the same name be used throughout From version 4 there is only a single *.int file produced, containing all the data concatenated.

name of the device (eg ‘*Income*’). To view the distribution, click on the cell containing the name (eg ‘*Income*’) and the cumulative and density distributions will be shown if a generated file exists.

Please note that it is inappropriate to define a distribution for ‘*Profit*’ as this is an output, and any definition will be ignored. The DSI methodology can in principle handle any probability distribution, and a selection of distributions have been implemented in the software. These are described in Appendix 1.

8.2.3 **Propagating probability distributions**

The final part of the computation process is to run the probabilistic algorithm that combines the two input probability distributions using the operator, and produces the output distribution. This is done by returning to the ‘Chart’ tab sheet, and calling up the pop up menu with a right mouse click on the operator block. Select the propagate item. This completes the probabilistic mathematics process and saves the results to a file with the same name as the output (eg ‘*Profit’*). The probability chart will be shown at the end, as per Figure 8.3. The chart may be panned or zoomed with right and left mouse clicks respectively.

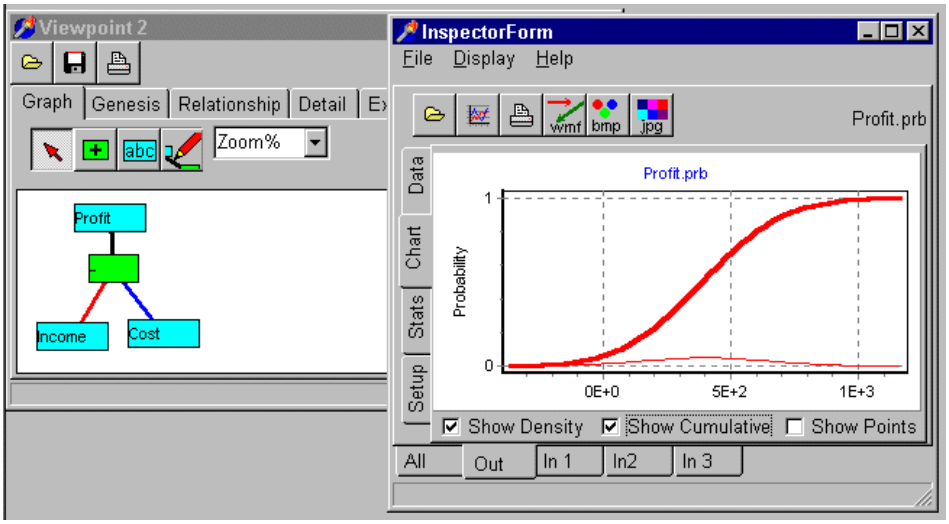


Figure 8.3: Propagation involves taking the input probability distributions and combining them using the given relationship (minus in this case) using the DSI probabilistic algorithm, to produce the output distribution. The block diagram graph at left shows the computation group, and the chart at right shows the resulting probability distribution for Profit.

The probability chart for any device on the graph may be shown by a right mouse click on the device. If nothing is shown then it means that the file has not yet been generated or propagated.

8.3 **Using DSI for managing uncertainty**

Once a system has been modelled in DSI, then the results may be used to identify sources of uncertainty and to assess the risk in the system. This is valuable information in deciding how to manage the project and its risks.

8.3.1 **Interpreting the results**

Figure 8.4 shows the density and cumulative distributions[137] for 'Profit'. The density provides a quick visual feedback of where the results are expected to lie, and how much influence the upper and lower tails have. A tail that is either heavier or more extensive in range indicates greater uncertainty and therefore more risk of extreme outcomes being achieved. The peak of the density is called the mode. For a symmetrical distribution like this the mode, mean and median are all the same, but this is not so in general. The mean is only obtainable by further calculation, as it is the average value. This calculation is available on the 'Stats' tab sheet. The median is determined from the cumulative distribution.

Here the output distribution looks very similar to a Normal distribution, and in fact it may be shown that this is the case since both the inputs were Normal and a simple operator (plus or minus) was used[138]. The mean and standard deviation from the 'Stats' tab sheet are therefore accurate statistics for the output. However this is not case in general even if a mean and standard deviation may be calculated.

[137] Probability distributions can be overly intimidating to interpret as most people do not encounter or use them regularly. However they are relatively easy to understand once some basic principles are established. Firstly, there is density and a cumulative distribution for each probabilistic parameter. The density is like a histogram, and its peak shows where the results are most likely to lie. The cumulative is the sum of the density from negative infinity up to the point of interest. It ranges from 0 to 1.

[138] An important theorem in risk simulation is the *Central Limit Theorem*. It describes the situation where a large number of variables are drawn independently from the same distribution, and it states that the mean of those n variables will be approximately Normally distributed, with a standard deviation given by $\sigma/\sqrt{n}$ where σ is the standard deviation of the parent distribution. The significance of this theorem for risk modelling is that variables that are added together will tend to produce a Normal shaped distribution. Similarly, the product of many variables tends to produce a log-normal distribution (Vose, 1996).

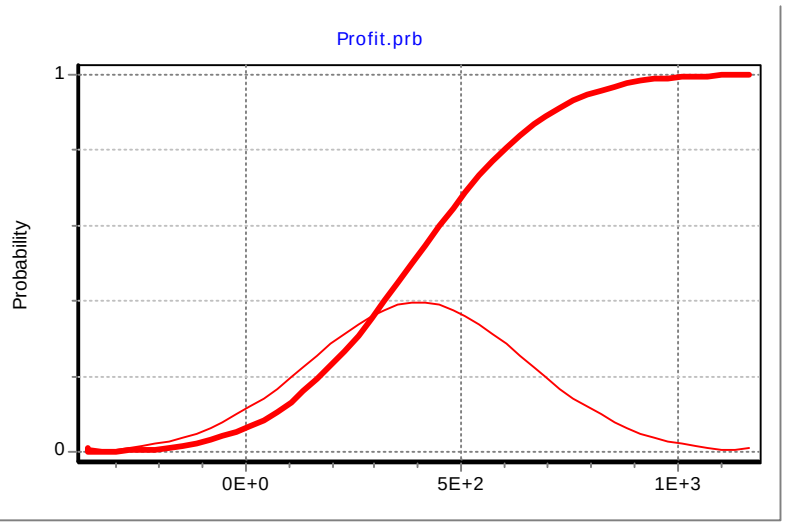


Figure 8.4: This chart shows the density and cumulative distributions for 'Profit'. The density is the lighter curve. The density has been artificially enlarged in height to make it clearer.

The cumulative distribution shows the probability of a result up to and including the given value. For example there is a 6.25% probability of an outcome from negative infinity up to zero, all of which values would be interpreted as loss. The value of the cumulative distribution is that it shows the extreme outcomes and how likely they are. The median is the 50% cumulative probability: half the time the outcome will be greater than the median, and half the time less than it.

8.3.2 **Extending to additional levels**

The graph-based approach of DSI permits the computation to be extended to deeper levels. There are two ways to do this, either to create more detail on the existing viewpoint, or by defining additional viewpoints and linking them together.

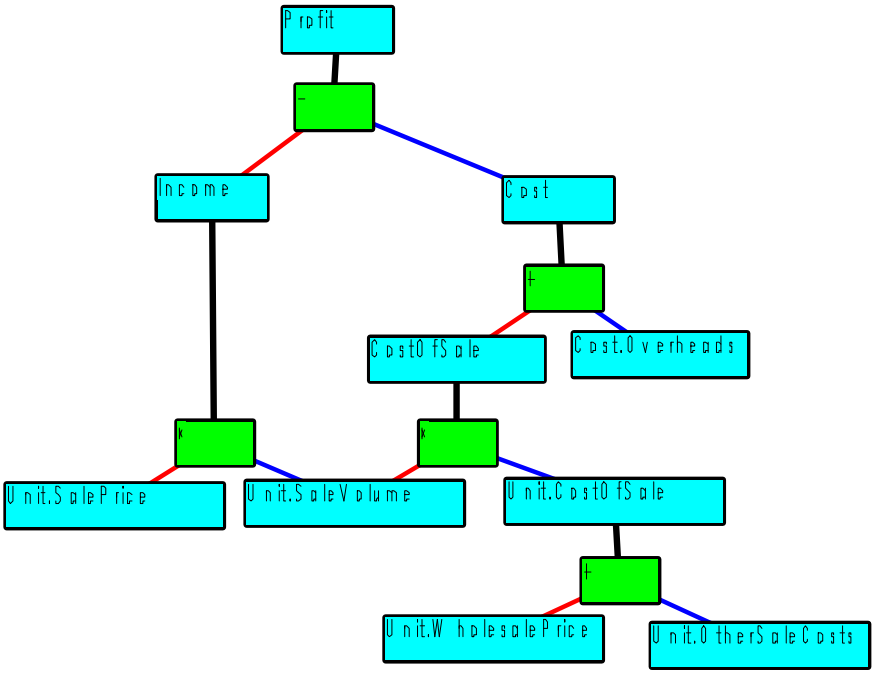


Figure 8.5: Deeper levels of detail may be created as the model is further developed. This graph extends the previous example by adding sub calculations to replace the parameters that formerly were assumed to be given distributions. Note the use of additional mathematical operators such as product (*) and plus (+). The model may be extended indefinitely.

Creating more detail on the existing viewpoint

The computation tree may be readily extended by adding more computation groups (output, inputs and an operator). The user achieves this graphically, by placing the components on the canvas using the mouse. An extended tree is illustrated in Figure 8.5. DSI can accept a parameter that is input to multiple groups. This is illustrated by '*Unit.SaleVolume'* in the figure.

Defining and linking additional viewpoints

For complex graphs it may be appropriate to open another viewpoint, and create it there. Devices that need to appear in both viewpoints should be given the same name. The device genesis may be described in one or both viewpoints, but whichever one was last generated or propagated will be used since usage is file-based. The ability to create additional viewpoints may be used to model sub trees,

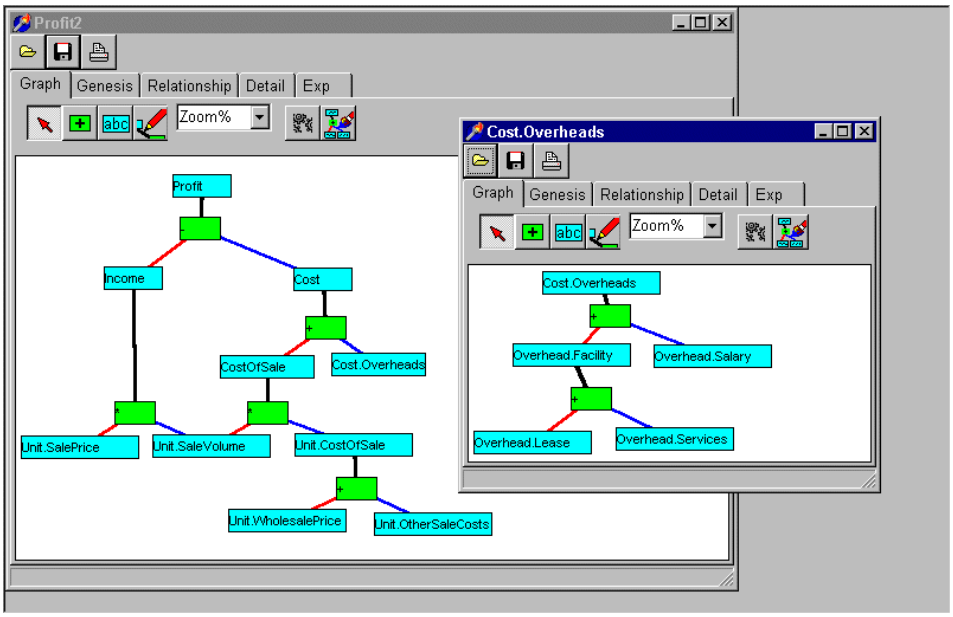


Figure 8.6: Linked trees may be set up in DSI. In this case the top level graph (left) contains the entry 'Cost.Overheads' and this is a separate sub tree which shows additional detail (right). There is only one probability file for 'Cost.Overheads', and it is accessed by both graphs as needed.

so that top level trees may be kept simple and easy to follow. This application is illustrated in Figure 8.6, where one parameter ('*Cost.Overheads*') is further detailed in the sub tree. This facility may also be used to model totally independent attributes of the system. For example one viewpoint (or set thereof) could be devoted to cost, and other viewpoints dedicated to reliability or machine performance.

8.3.3 Modelling qualitative relationships

When relationships are qualitative then mathematical operators lose their meaning. Qualitative parameters are typically described with text.[139] Instead it is necessary to use mapping relationships. These are set up initially using the Integrity-T tool within the DSI software, see Figure 8.7.

[139]The interested reader is referred to Chapter 6 for a description of how the terms quantitative and qualitative are used in this work.

Qualitative map relationships are especially useful when the underlying mechanisms in a system are unknown. In these cases the map relationships permit expert opinion to be used instead, and they require that the expert specify how certain or otherwise the opinions are. Maps have been described in more detail in Chapter 5.

8.3.4 **Disseminating results**

The DSI software provides several ways in which results may be distributed. Graphs and charts may be printed or saved in a variety of file formats. Windows metafiles (wmf and emf) may be produced, and bitmaps and jpg files. The metafiles are recommended for presentation purposes as their resolution increases as they are scaled up in size, unlike bitmap and jpg which are fixed.

Risk Maps are a communication tool provided by the DSI software. They permit the entire model to be viewed over the web. The ‘web’ button on the Print tab sheet starts the process. The software first creates an image of the graph, and then it creates images of each probability distribution. Finally it creates an *image map*. The result is that the graph appears in the users default web browser, and clicking on a block in the image loads up the relevant probability distribution or sub tree. An example of a Risk Map is shown in Figure 8.8. It should be noted that all the Risk Map images are static, and cannot be changed except by repeating the process from DSI. The Risk Map provides the means to communicate a model to others in a familiar web medium, permitting them to drill down to the level that

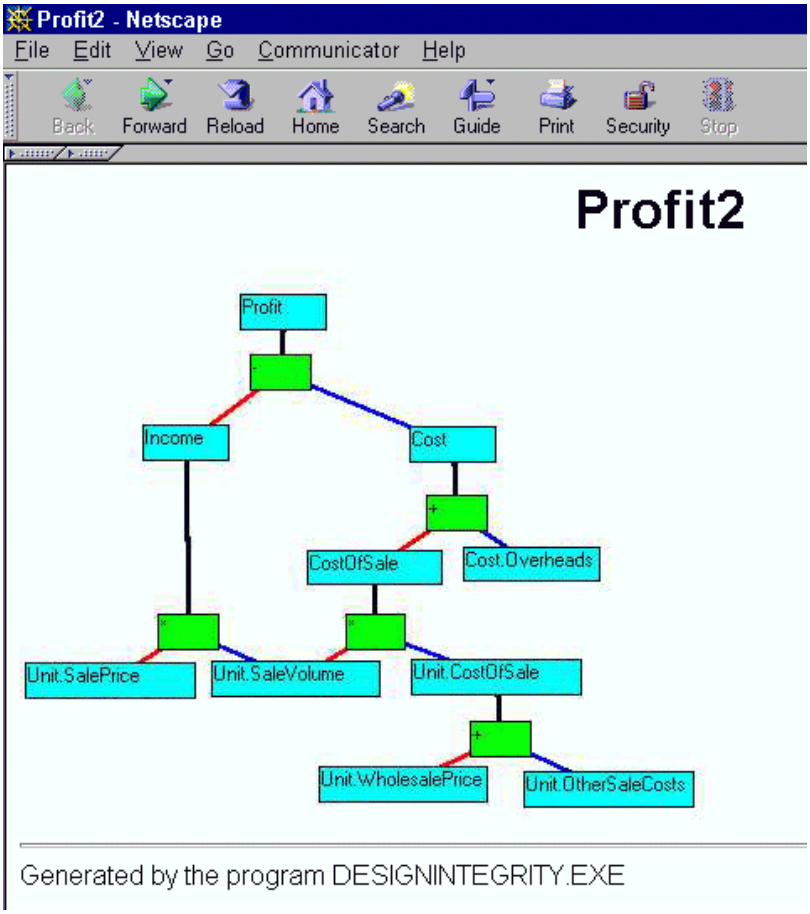


Figure 8.7: DSI is able to automatically create web based image maps such as this. These maps may be viewed with any browser. A mouse click on a blocks loads an image of the probability distribution for that parameter (or a sub tree if appropriate).

concerns them without having direct access to the original data.

8.4 **Conclusions**

The DSI software provides tools whereby a computational graph may be created and probability distributions propagated through it. The method accommodates both qualitative and quantitative relationships and data. The software implementation provides a standard Windows interface for ease of use.

Chapter 9

Software development

This chapter describes the development of the Design for System Integrity methodology into software. The validation of the software is described.

9.1 **Strategy for software development**

The Design for System Integrity (DSI) methodology involves considering multiple viewpoints of a design, and simulating results under conditions of *process variability* and *uncertainty of analysis*. In addition, physical devices and properties in the model can be substituted with new data, and the effects in that and other views determined. A bland statement of the intent of the methodology would have been insufficient without showing how such intent could be realised, if at all. It was therefore necessary to develop a software system to explore the feasibility of the methodology. The successful development of a software system demonstrates that the methodology is indeed feasible. The software allows the methodology to be scrutinised, challenged, and further explored.

A software embodiment is essential as the methodology is too computationally demanding to be practical as only a paper-based system. The software algorithms that perform the quantitative (Integrity-N) and qualitative (Integrity-T) computation are described in previous chapters. The whole is tied together in a software application that provides a graphical user interface and runs under Microsoft Windows 95+ or NT. While the software itself is a routine application of computer science knowledge and is therefore not listed in this document, it is essential for embodying the methodology. It was also a significant labour undertaking and it would be appropriate to record the major challenges faced in the development.

The whole application was developed in *Delphi*, which is an object oriented programming (OOP) language from Borland-Inprise. Delphi was selected because of its strong mathematical capabilities (provided by the *Pascal* language inside it) and its excellent tools for creating a user interface. Delphi also generates a fully complied executable. This is important as it provides vastly superior computational speed compared with interpretive compilers (eg Visual Basic) which compile line by line on the fly. Also important, the executable files generated by Delphi are truly

standalone and do not require the presence of Delphi on the user's computer, again unlike Visual Basic. Finally, there are no licensing issues with Delphi.

Delphi provides a blank form with various ready-to-use components such as buttons, memo boxes, panels, save/open dialogs, and grids to name a few used here.[140] The programming effort is then to place suitable components on a form and customise them, and then to write the code that needs to be executed for certain events (eg when a given button is pressed). The code writing effort was easily the greater, as this application makes heavy use of mathematical functions and arrays, file handling and graphics. The software consists of over 30 000 lines of code written by the developer. Additional code was generated automatically by Delphi. All the programming on this project was done by the author.[141]

Building the application from the ground-up provided the ability to add special features, which would not be the case if a shell (eg expert system or relational

---

[140]Creating a blank form involves Delphi creating two files. The first has a dfm extension and contains the windows instructions for creating the form and its buttons etc. A simple blank form would contain data such as:

```
object Form1: TForm1
  Left = 192
  Top = 107
  Width = 544
  Height = 375
  Caption = 'Form1'
  Font.Name = 'MS Sans Serif'
 .....
 end
```

The developer does not edit this file. The second file is a Pascal (pas) file containing the actions of the software, and these are coded by the developer with some minor assistance from Delphi. For example, if at initial design time the developer double clicks on the blank form, then Delphi adds the following default Pascal code:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
end;
```

This event handler of itself will do nothing, and the developer has to add code between the 'begin' and 'end'. For example if he added '*ShowMessage('Hello');*' then when the application was compiled and run, and the blank form clicked, a message box would appear displaying 'Hello'. Compiling a project creates one executable file from the dfm and pas files.

[141]Acknowledgement is given to Jason Butler (then at Fisher & Paykel, New Zealand) for initial guidance on use of 'shapes' (though the final implementation did not use them), and to the Delphi user manual, Cantu (1998) and various Internet pages for tips.

database) were used. Against that must be set the disadvantage of having to programme in all the computational and user interface code, which required significant effort and the necessary software development skills. In this chapter are described some of the more significant development features of the software.

As an OOP programming language Delphi makes it possible to define new *classes* of variables (including visual objects like buttons) by inheriting properties from *ancestor components* and adding more properties, methods and procedures. However Delphi has an extensive library of ready to use components, and therefore it is only necessary to define new classes if (a) there is no suitable existing component, or (b) a component is being written for other programmers to use. ‘The difference between component writers and application developers is that component writers create new classes while application developers manipulate instances of classes’ (Borland Delphi 5, 1999, p 32-1). The development of the DSI software was the development of an application. It was unnecessary to define custom classes as the well stocked Delphi library (VCL) contained enough to write the application using standard classes. This makes it possible to achieve more programming outcomes than if everything had to be developed from first principles. The code is also likely to be more error free than if custom classes were developed.

9.2 **Programme structure**

The structure of the program is shown in Figure 9.1 as a flowchart. This is a simplified representation of the primary functionality, and there are many other features of the application that are not shown. Events or actions are shown in the figure as blocks, and outputs are in text without a block. The events correspond to blocks of code in the software. The events are numbered and a brief interpretation follows.

The first event (1) for the user is to create the graph[142] by drawing with the mouse or by loading an existing file. The result is a graph on the screen, and entries in two grids, the relation grid and the genesis grid. These grids are on other tab sheets of the form. The user can also print or create an image of the graph at this point (5) or save it to disk (6) in text format with an ‘int’ extension. The next important event in the process is generating the input distributions (2), when the user provides the distribution parameters in the genesis grid. The outputs of this event are probability files on disk. There is one file for each input distribution, and the files are in comma separated variable (csv) text format for easy integration into spreadsheets or word processors. They have a ‘prb’ extension. Next the user runs the simulation (3) which produces output files on disk in the same prb format. These files may then be inspected (4) and statistics measured or images taken for presentation purposes. If the shape of the result suggests that the distribution is ill conditioned, say in the tails, then the user can check this (7) and modify the genesis parameters (2) to more adequately control the tails.

The above description provides an overview of the software functionality, with the events describing the actions performed by large blocks of program code. The discussion now focusses on the detail in blocks 1 and 3.

[142] The term graph as used here refers to the network-like representation of the problem, whereas chart refers to a plot or histogram graphic.

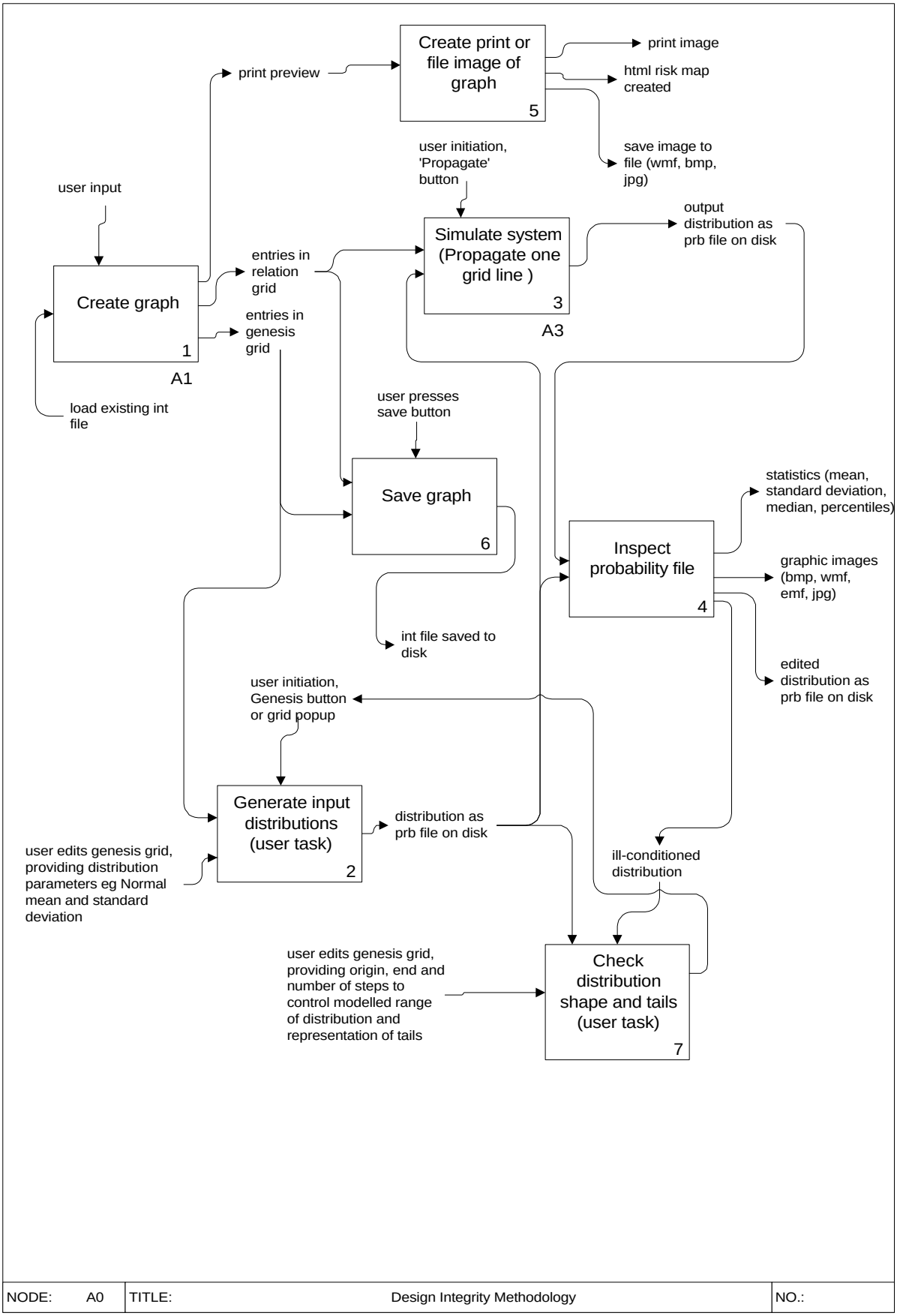


Figure 9.1: Overview of DSI method.

The event ‘Create graph’ (1) has a detailed diagram as indicated by ‘A1' below it. These details are shown in Figure 9.2. The action of creating a graph on the screen involves windows programming. Fortunately Delphi simplifies this as it insulates the software developer from direct contact with the lower level Windows programming interface functions, but there is still a substantial amount of code that the developer has to write. The graph is created by using the mouse. First the user selects the drawing mode (2). If it is to create either genesis or relation panels then the subsequent mouse down event (1) results in the genesis or relation panel being created. It is important to note that the panel is a Windows component that has to be deliberately created by the developer’s code since this occurs at run-time rather than at initial design-time, i.e. neither Delphi nor Windows automate this process. This adds to the programming challenge.

If the user has selected the arc drawing tool then a mouse down event on a panel (4) is the next appropriate event. The panel refers to the blocks, which could be either a relation (eg +, -) or a genesis device (eg ‘Pump.Cost’). The primary output of this event is a line painted between the two panels that have been clicked on. However this is not a trivial exercise as different types of line must be painted, depending on the type of link being made, or whether a link is valid at all. The relation string grid is also updated once an arc has been accepted for painting. Other outputs of the mouse down event on the panel are dragging the panel to a new position on the screen, or activating a pop-up menu (3) with a right button mouse click. The pop-up menu provides the user with several choices, and therefore has the various outcomes shown. Event A13 is involved and is not shown here or discussed further.

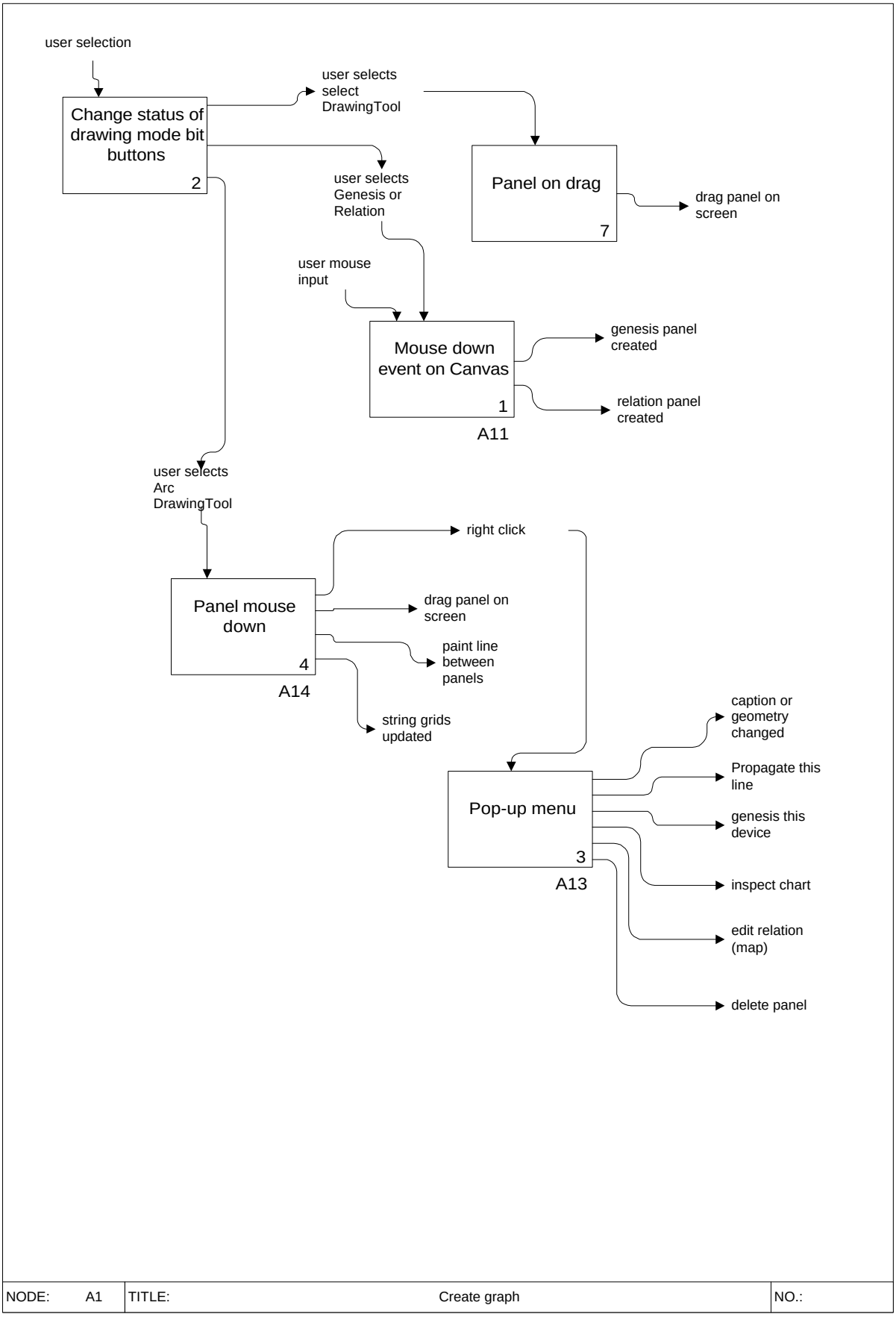


Figure 9.2: Details of create graph event

For those readers who are interested in the detailed software implementation the Figures 9.3 to 9.7 provide additional detail.

Action A11 is given in Figure 9.3 showing the manner in which the mouse down event creates the panels. Figure 9.4 shows action A112 and Figure 9.5 action A113.

Action A14 is given in Figure 9.6 showing the effect of a mouse down event on a panel. Figure 9.7 gives details of the left click event on a panel.

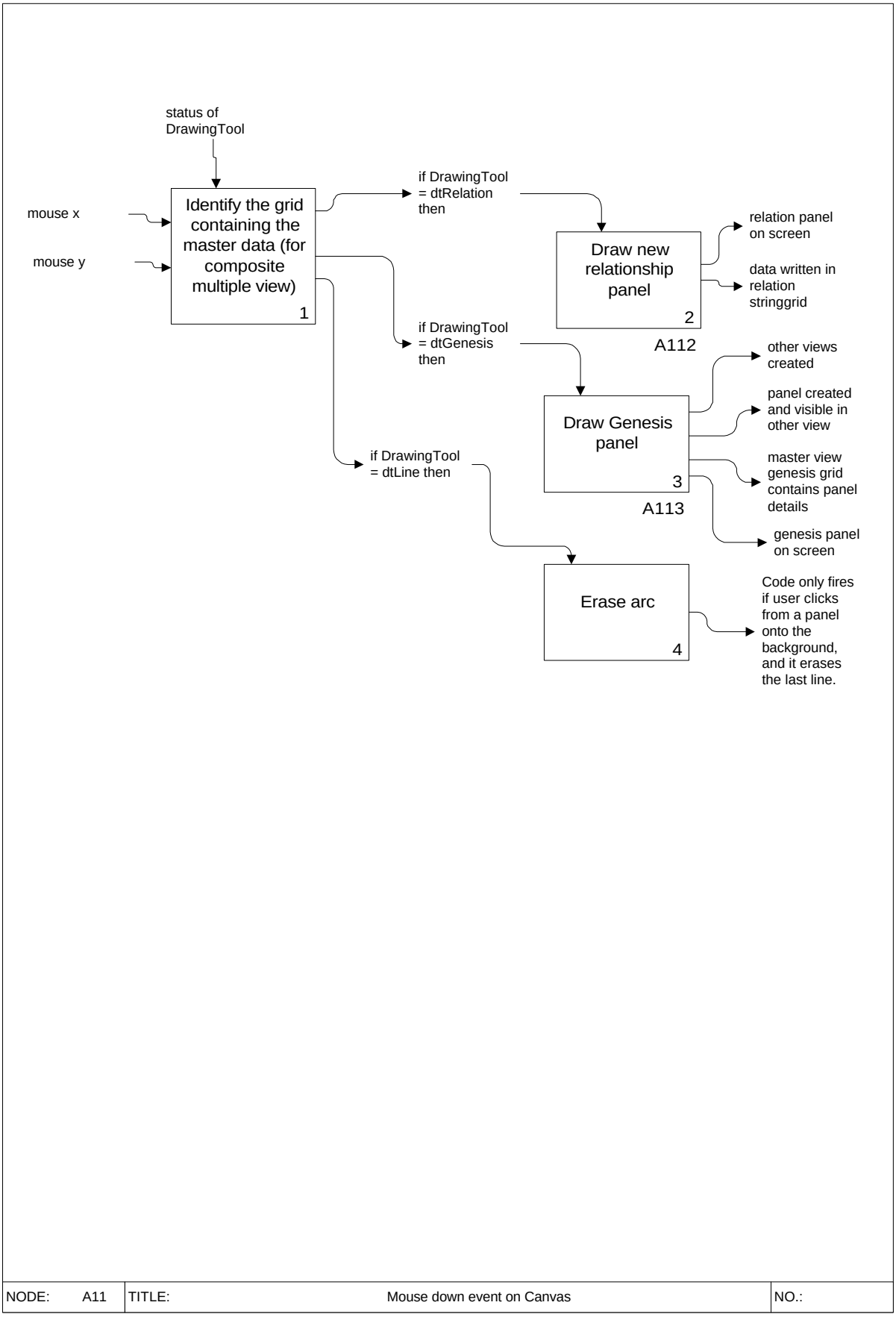


Figure 9.3: Details of mouse down event

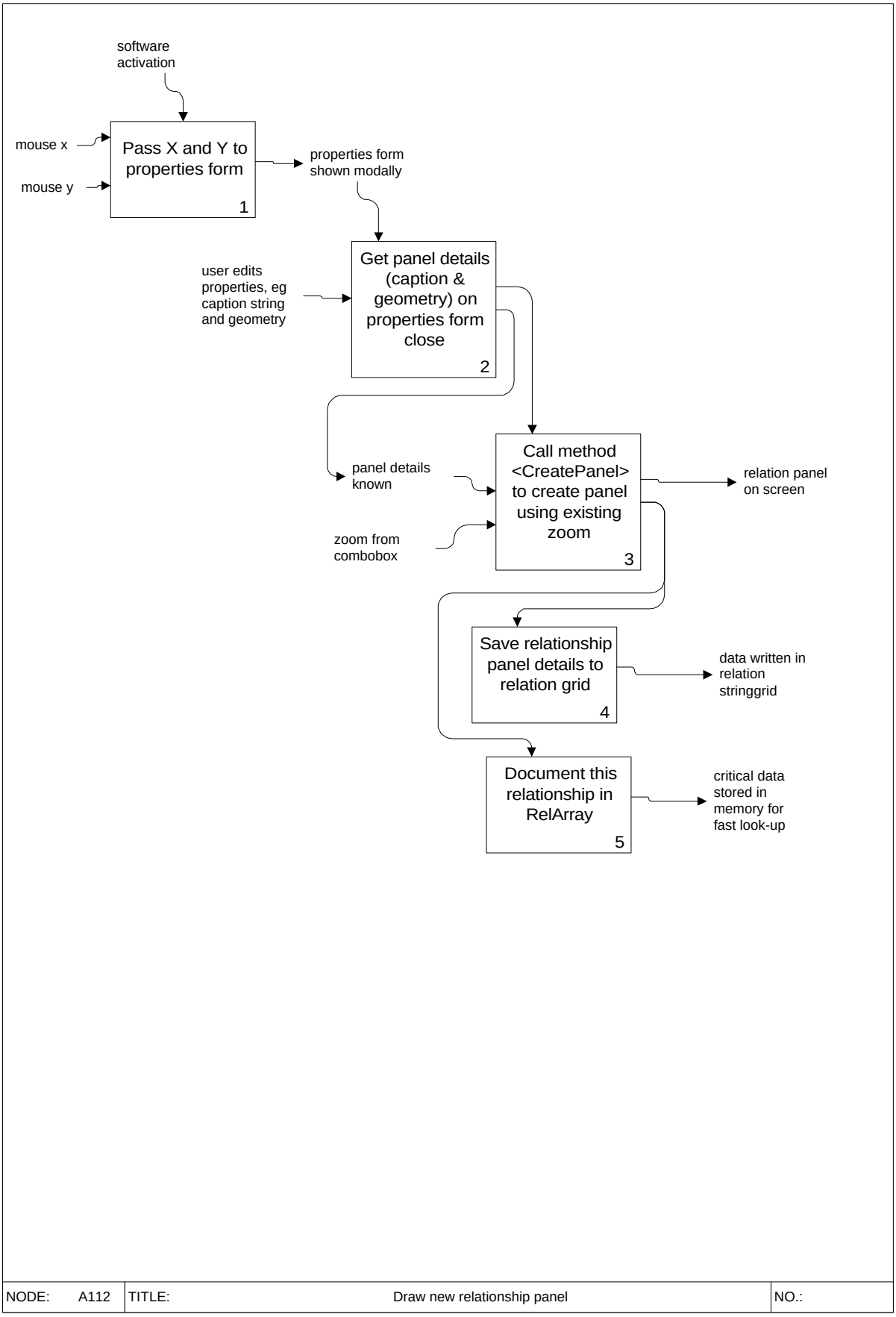


Figure 9.4: Details of how new relationship panel is drawn

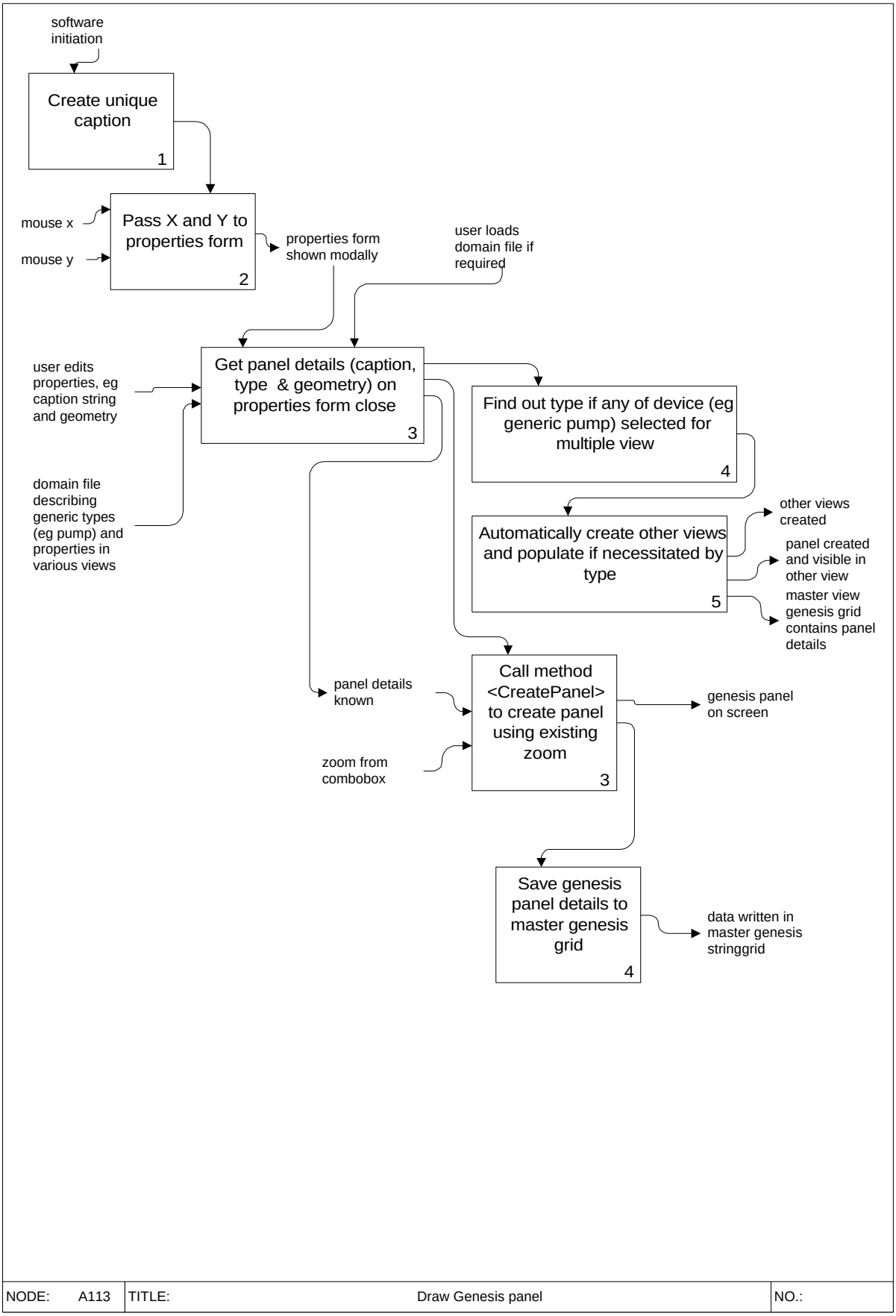


Figure 9.5: Details of how new genesis panel is drawn

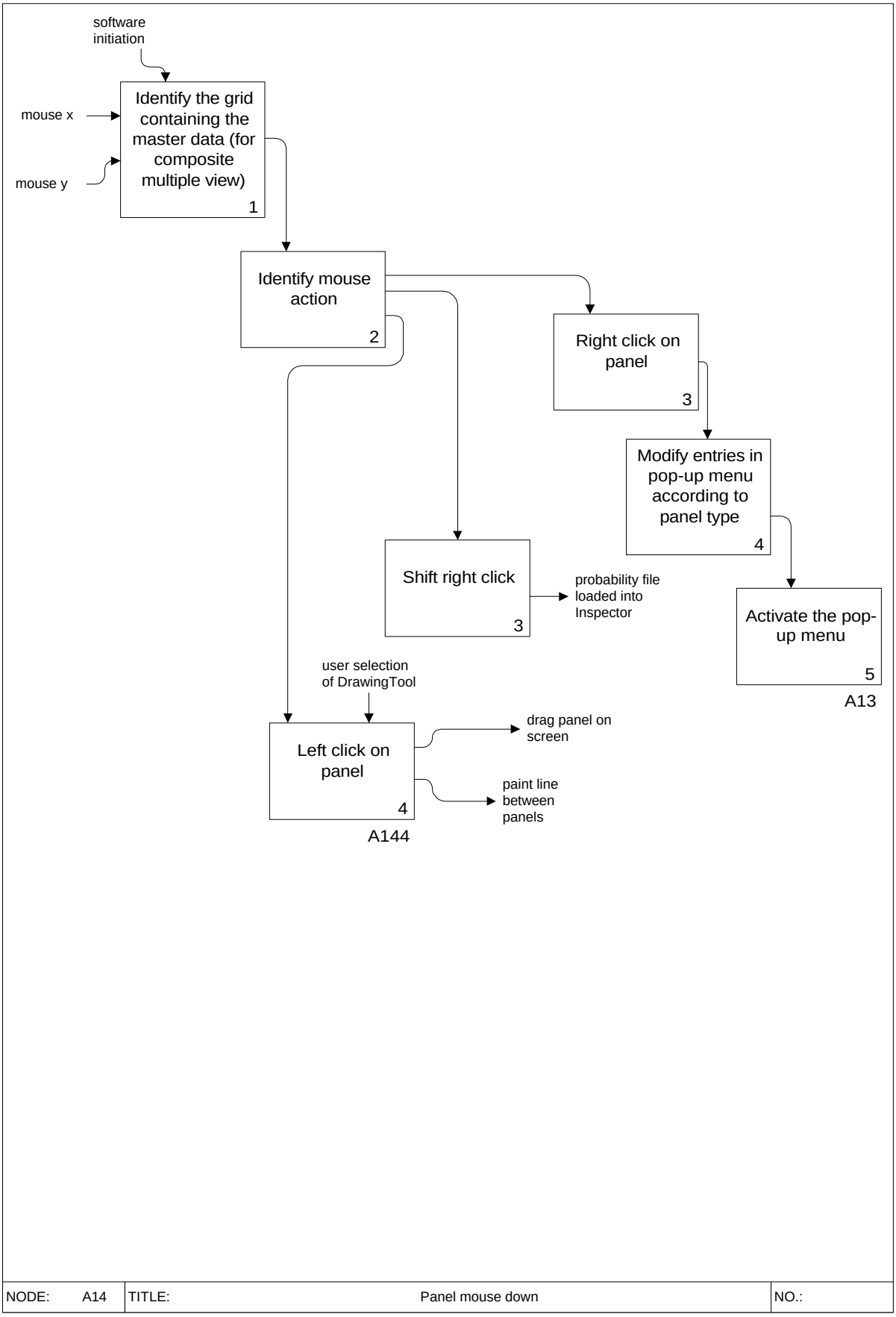


Figure 9.6: Details of mouse down event on a panel

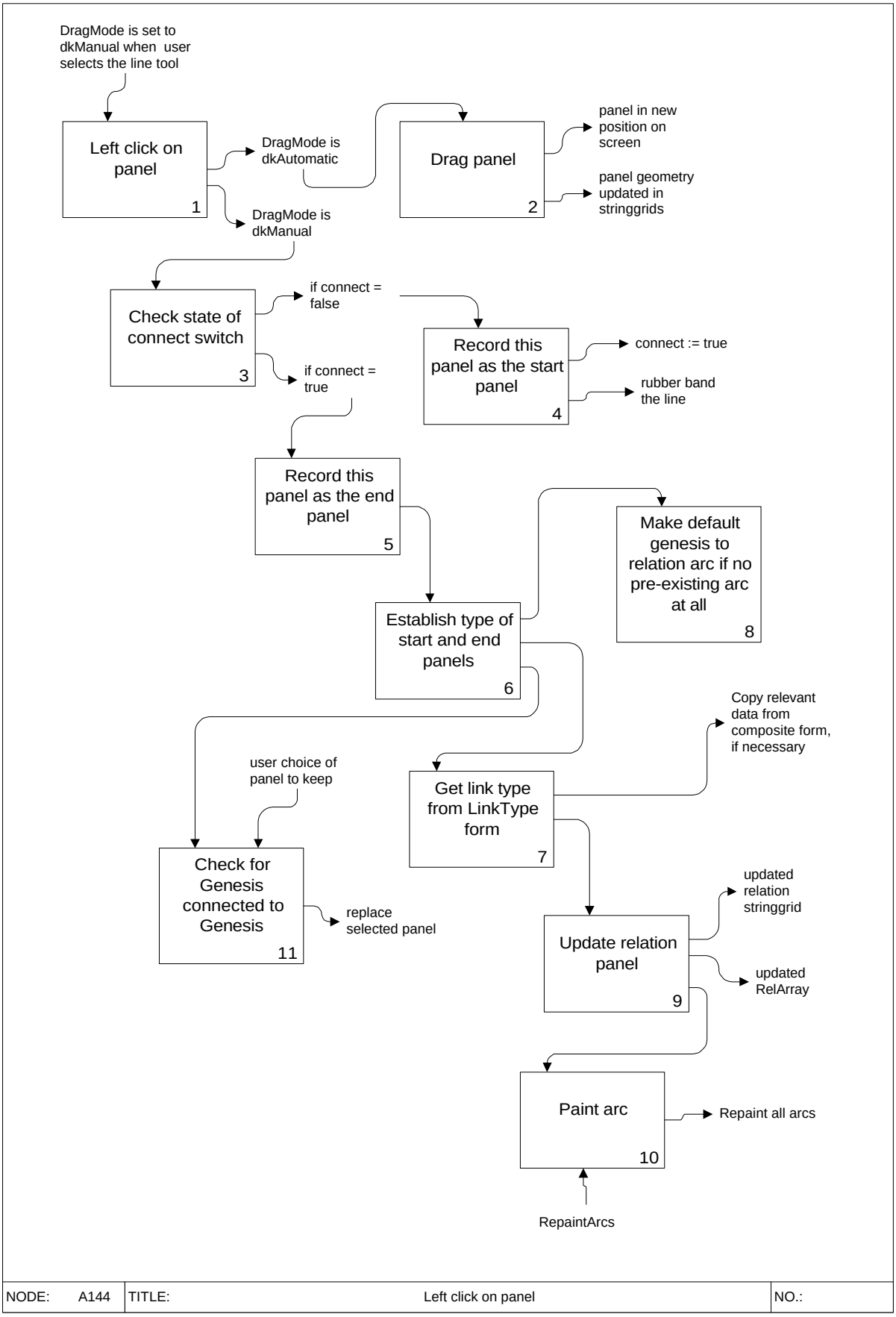


Figure 9.7: Details of left click event on a panel

Discussion returns now to the action called ‘Simulate system A3’ which is described in further detail in Figure 9.8. This is the event that applies the probabilistic computation, both quantitative and qualitative, and is represented in the software by a method called ‘PropagateOneGridLine’. On calling this method it finds the appropriate line in the relation grid and gets the file names and other data (1). It also gets any alarm limits from the genesis grid (2), and then checks the operator (3). The textual analysis Integrity-T is run if the operator starts with ‘map’ (4) , and this involves loading the map to another form (5), loading the assert files (6), running the Integrity T algorithm (7), saving the results to file (8) in prb format, and checking the alarms (9).

If the operator is a recognised quantitative one (10), then the system loads the assert files to memory (11), creates a blank output file (12), runs the Integrity-N algorithm (13), saves the results to file (15) also in prb format, and checks the alarms (16). Although the actions and outcomes are similar for the qualitative and quantitative cases, the code is different and hence the repetition of these actions. A Monte Carlo simulation (14) is provided as an alternative quantitative process.

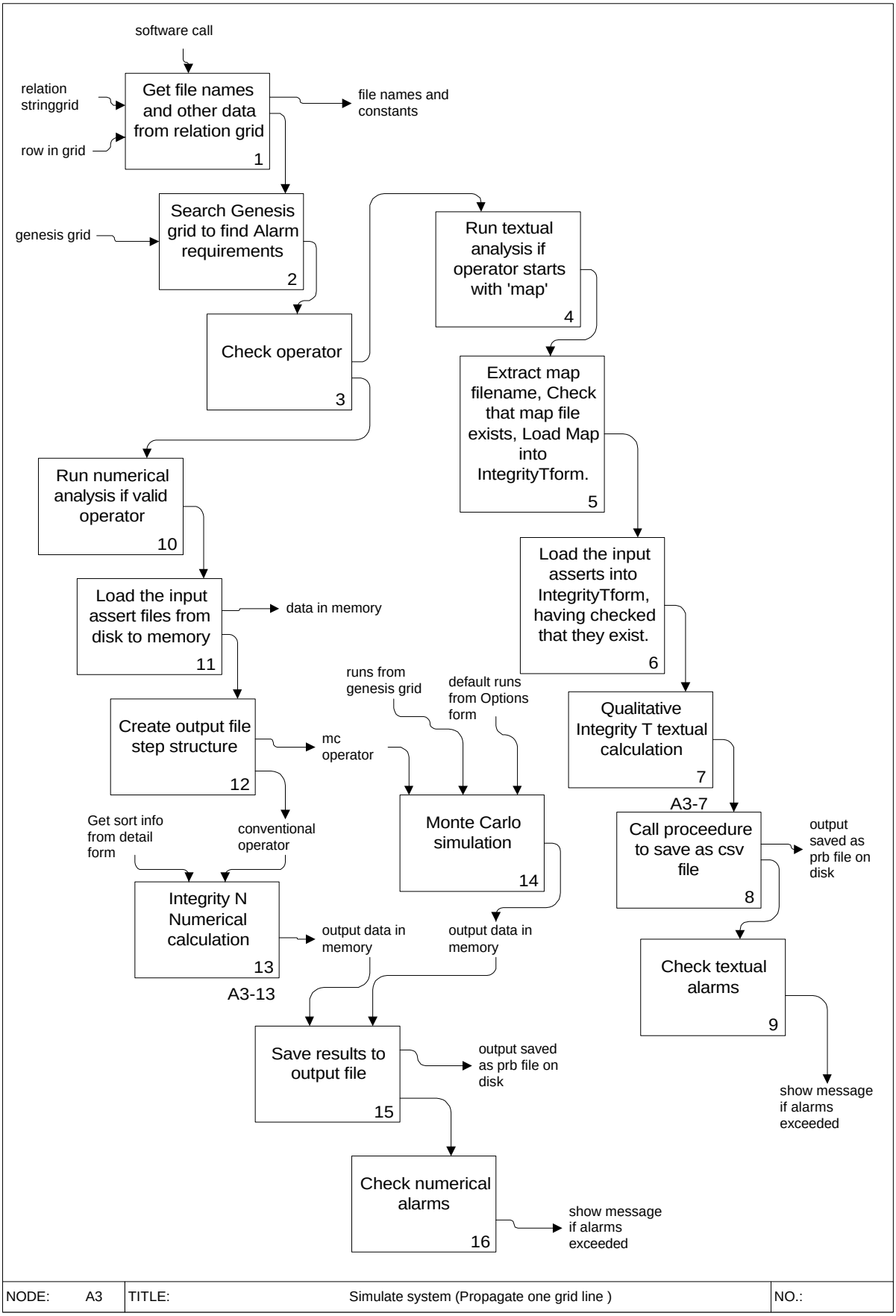


Figure 9.8: Details of simulate system event

The operation of the qualitative Integrity T calculation[143] is detailed in A37 in Figure 9.9. The first action is to load the map from the grid into memory (1), then to load the assert data (2), create a blank output file (3), check input labels (4) and check the file type (5). Numerical input files use a proportionally binning process (6) whereas textual files assign the whole input probability to the corresponding map label (7). The joint probability is then calculated for all combinations of inputs (8), sorted into bins (9) and written to the grid (10).

[143]This is a procedure named ‘IntegrityTpas.CalculateJointT’ in the software.

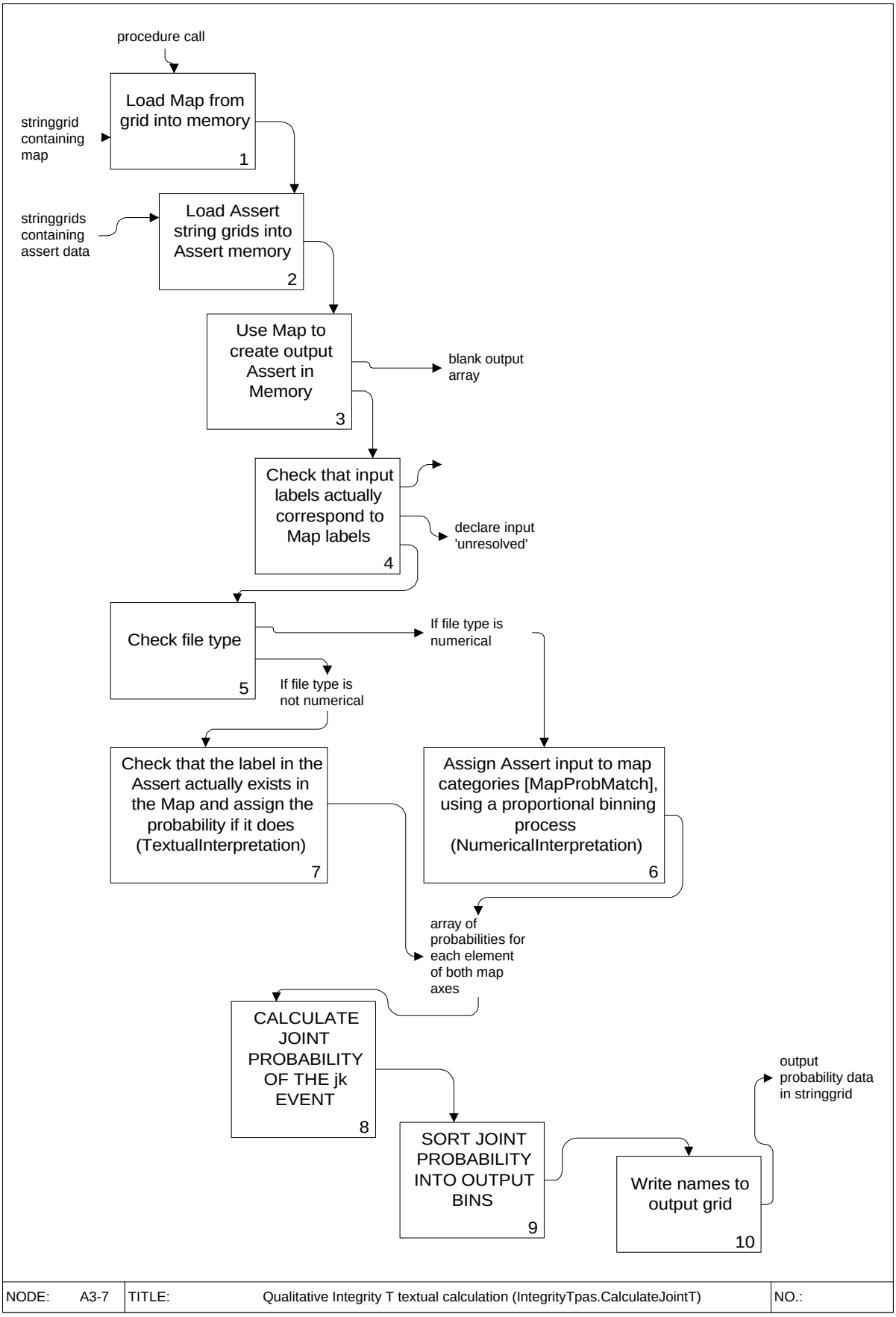


Figure 9.9: Details of qualitative calculation

The numerical quantitative calculation is detailed in A3-13 in Figure 9.10. Again this corresponds to a procedure, the main parts of which are initialisation (1), doing the combinatorial joint calculations (2) and sorting into output bins (3) with the end result being the output as an array in memory.

Summary

As these diagrams have progressed into further detail, so they become increasingly descriptive of the actual Delphi code, and in the limit they could be devolved down to actual programme listing. The diagrams selected for inclusion here are only those of the crucial sections, and most of the methods and code in the project have not been discussed. There is much other overhead and error trapping code in the project, but it would be tedious to discuss all this in detail and superfluous to the central premise of this work, namely that the DSI methodology can be embodied in a workable software form.

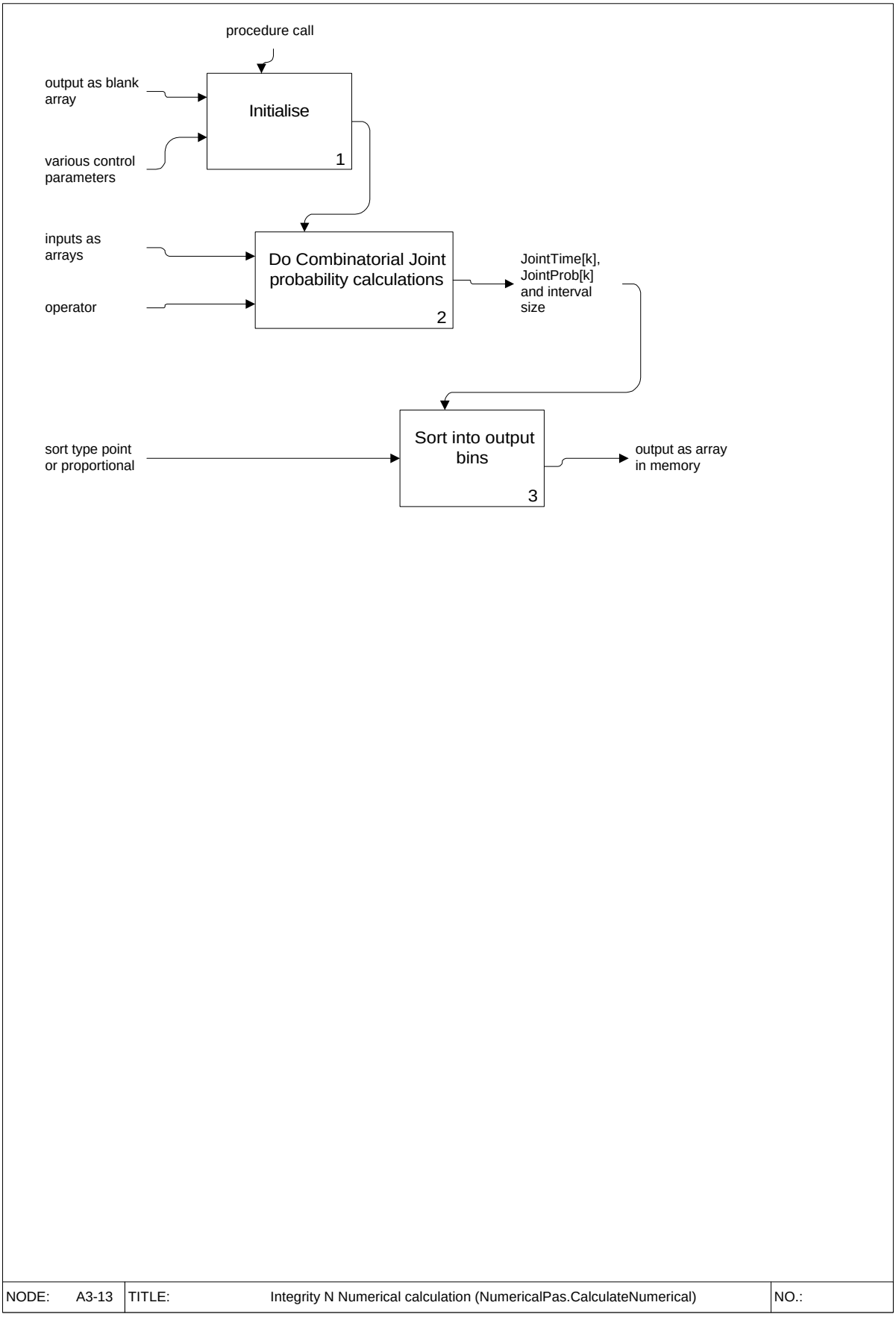


Figure 9.10: Details of quantitative calculation

9.3 **Software milestones**

The most challenging aspects of the software project included:

- Providing the mathematical functionality for the Integrity-N method. This algorithm involves extensive computation and array usage. It also required care to ensure that errors were eliminated.
- Providing the map (3D decision table) algorithms.
- Providing the software that conditioned quantitative and qualitative data for mutual interchange. The user interacts with some origin reconditioning software, but more code services the background needs and makes sure that the data are interchanged successfully. A significant task was ensuring that the file format supported this, especially of the probability and relationship files.
- Ensuring that the above algorithms were suitably reliable and robust against various user inputs. This involved creating software checks for validity against complex conditions, and protecting code so that execution could collapse gracefully if necessary, advising the user where possible.
- Providing alarm function for parameters that had gone out of user selected bounds. This involved getting and storing the user bounds, then checking them during execution, and alerting the user as necessary.
- Providing direct file input and output access. Though Delphi provides the base tools necessary to read and write a line from a file, their use is not well documented either in the manuals, other books, or on the web. This is because most software that is now written uses databases, which handle all the file access for the programmer. Delphi's support for databases is excellent, but the direct file input-output (*IO*) is neglected in terms of documentation. In this project the author decided to go the route of stand alone files, because of concerns about large and cumbersome databases if all the data were in one file. This choice meant the author had to handle all the file IO, but also the deployment of the software is much simpler, as a database does not also have to be installed, nor a database engine. Simply copying the executable file is all

that is required. Dynamic link libraries (DLLs) were avoided for the same reason.

- Providing a graphical user interface whereby the user can draw a graph by creating panels on the form and connect them with lines. This required significant effort as the programmer has to take ownership of creating all the arcs and graphics and ensuring that they are repainted under the right conditions. In the project as originally envisioned a very much more modest graphical user interface was to be provided, and the early versions of the software only went that far. In that form the software provided the basic functionality but required skill to operate. The more extensive interface was developed when it became apparent to the author that the provision of a better interface would be useful in making the method more accessible to others.
- Extending the graphical user interface to multiple views (technically these are multiple document interface (MDI) child windows) required significant work, especially the creation of devices in other views in the background. The interaction with the domain file and providing the substitution capabilities was also demanding.
- Providing printing. Curiously, the Delphi 5 user manual does not include the word ‘print’ in any form either in the contents or index. This made implementing print facilities difficult. However Delphi does provide print functions even though poorly documented, and Cantu (1998) had documented sufficient of them to get the author on the right track.

Possibly the greatest efforts were required in creating the Integrity-N method, creating the Integrity-T method, ensuring the robustness of the software, providing the graphical user interface, and extending the interface to support multiple views with automatic device creation and substitution capability.

Unlike other systems that are shells waiting for the data to be put in, Delphi is a blank canvas. The disadvantage is that this requires extensive programming as Delphi only provides loose bits and pieces such as buttons with blank methods behind them. Features such as the ability to drag the devices of the on-screen graph had to be

coded by the developer to a significant extent, in addition to the probabilistic computation algorithms and error checking. The major advantage of using a powerful software system like Delphi is that it gives the developer full control over the project and minimises constraints on what is possible. If a shell such as an expert system or a relational database had been used, then the project would have been constrained by the capabilities of that shell: if features were required that were not in that shell, then either they would have to be foregone, or the source code for the shell would need to be obtained and extended where possible. Also, the shells present a fixed user interface, which though it may be customised cannot be radically altered. Using Delphi avoided all those constraints, even if at the cost of greater development labour, and made it possible to develop an application to meet the specific intent of the proposed DSI methodology. It was also possible to extend the capabilities mid way through the development to meet unanticipated requirements, and this might have been difficult had the project been locked into using a particular shell. Delphi was an excellent tool for developing the DSI methodology and has been a crucial enabling factor in its success.

9.4 **Using Confidence intervals to express process variability**

Additional methodologies and software applications were developed to support the main DSI system. This section describes a mechanism for a user to express confidence in estimates. The method is applicable to modelling quantitative uncertainty.

The nature of decision problems is that they often have limited information. Walley et al (1990) note that is important in such cases to model both uncertainty and indeterminacy in the key variables, and to use expert opinion and any other available information about the variables. They feel that lack of information and disagreement among experts should be reflected in imprecise probability assessments.

The issue of measuring prediction confidence is discussed in artificial intelligence and in expert systems in particular. The interest is in being able to provide the user with some indication as to the accuracy of the predictions made by the system. Similar needs arise in decision analysis. The tools that have been used to deal with confidence are uncertainty factors (expert systems), fuzzy logic and of course statistical confidence levels.

9.4.1 Defining subjective probability

A subjective assessment of uncertainty is commonly encountered in all types of decisions. *Decision analysis* and related tools can provide means to approach subjective probabilities in a systematic manner. *Subjective probability* is the extent to which a personal *belief* is held. Phrases like “often” and “usually” are commonly used to show the extent of the belief, but the problem with verbal descriptions is that they mean different things to different people. There is a need to assess beliefs in a more systematic manner. Decision analysis accomplishes this in several ways. The simplest method is to assess the probability directly by asking, “what do you believe the probability of this event to be?”. However this approach may be difficult to implement when the concept of probability is unfamiliar to the person. In such cases there is an alternative method which is used. This is to use hypothetical bets to explore the personal beliefs: find amounts X and Y such that the person would be equally willing to win X or lose Y on the outcome. Then the subjective probability is Y/(X+Y). There is a similar method that uses two hypothetical lotteries.

Biases

The assessment of subjective probability is relatively easily affected by bias (Vose, 1996). Common sources of bias include:

- *Representativeness*, also called *stereotyping*, whereby an item is judged to be part of a group because it displays some of the stereotyped characteristics of the group.

- *Availability*, the freshness of an event in memory such that similar subsequent events are dramatised.
- *Overconfidence*, and *underestimating* the probability of extreme outcomes. This may be caused by using existing data as a base (or *anchor*), from which to make predictions. One method to reduce the bias is to assess the extreme outcomes (eg 5% and 95% fractiles) *before* assessing the median (see below).
- *Motivational bias*, whereby the person assesses the subjective probability either higher or lower than what they actually believe because of some incentive (which may be subconscious).

9.4.2 Assessment of continuous probabilities

Decomposition is used in decision analysis to develop a model of the decision process. By considering the constituent elements of the larger decision, it is easier to provide individual probabilities that can be built up into the larger picture. The method is identical to that of fault trees in reliability engineering. The method of assessing probabilities needs to be sufficiently rigorous that the results can stand up to scrutiny. A cumulative probability distribution is difficult to put into decision analysis because it is continuous. A simplification that gets round this is to approximate the continuous distribution with a discrete one.

One such simplification is the *Extended Pearson-Tukey* method, which allocates probabilities according to the fractile, see Table 9.1.

Fractile	Probability
0.05 (5%)	0.185
0.50 (50%, median)	0.63
0.95 (95%)	0.185

Table 9.1: Extended Pearson-Tukey method

This particular method is suitable for symmetric probability density functions, such as the normal distribution.

Another method is using *Bracket Medians*. The cumulative distribution is divided into probability intervals (eg 0 to 0.2, 0.2 to 0.4, etc.). Within each bracket the median is found (for that bracket). The median means that there is an equal probability above or below that value. The individual medians are then used with the probability at the mid point of each bracket, and these make up the discrete probability distribution. More brackets give a finer resolution. The method may be applied to any probability distribution.

9.4.3 Use of confidence intervals for process variability

DSI applies confidence intervals, eg in terms of three discrete values of cumulative probability. Typical points might be the 5^{th}, 50^{th} , and 95^{th} percentiles, though other points are also possible. For example, the cost of a component might be defined as $5.00 median (50%) with 5% confidence that it would be less than $2.00, and 95% confidence that it would be less than $12.00.

Probability distribution

It is not necessary to limit the probability distribution to the normal. In fact the normal distribution is not ideal for many engineering measures. The normal distribution is symmetrical about the mean, and extends from negative infinity to positive infinity. In contrast many engineering measures such as mass, cost, fatigue life and geometric dimension have skew distributions and certainly do not permit values below zero. A distribution such as the Weibull might be more appropriate sometimes.

Method

Given that the user has provided three data points, such as the 5%, 50% (mean) and the 95% cumulative values, the next stage is to fit (eg) a Weibull probability distribution. The method for doing this has been established in the reliability engineering field. It is illustrated below with example data.

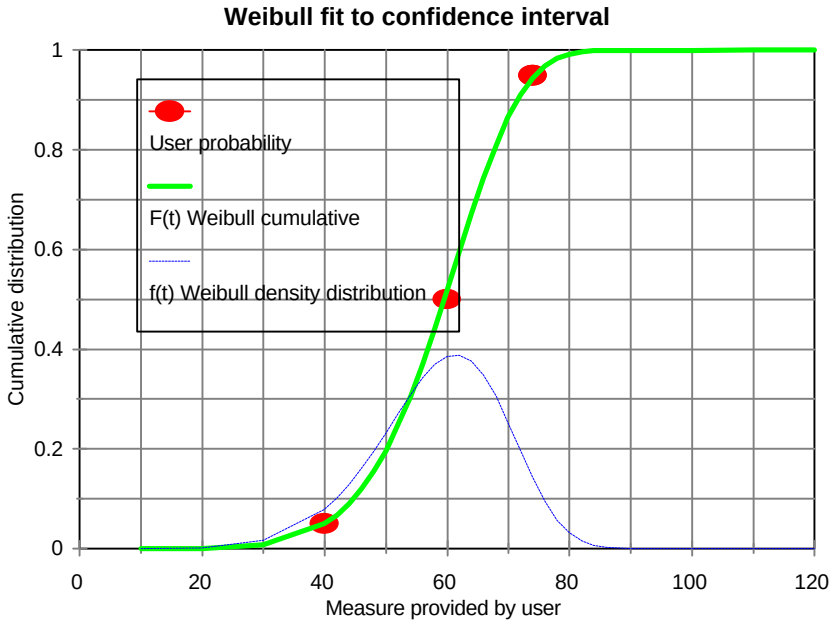


Figure 9.11: Confidence interval fitted by Weibull distribution. Oval markers show measures and probabilities provided by user, and thick line shows fitted curve (cumulative distribution). Thin line shows density function (not to scale) for comparative purposes.

(1) The user provides measures t and their cumulative probabilities F(t) as per Table 9.2.

Measure t	User probability F(t)
40	0.05
60	0.5
74	0.95

Table 9.2: User estimates

(2) Apply transformations x(t) = ln(t) and y(t) = ln(-ln(1-F(t))) to the data (see Table 9.3).

Measure t	User probability F(t)	x(t)=ln(t)	y(t) = ln(-ln(1-F(t)))
40	0.05	3.6889	-2.970195
60	0.5	4.0943	-0.366513
74	0.95	4.3041	1.0971887

Table 9.3: Weibull transformations

(3) Plot y(t) against x(t) and fit a straight line using a least squares. Determine the slope a and the intercept b as follow:

Slope	6.5859
Intercept	-27.2818

(4) Calculate $\beta = a$ and $\eta = e^{-b/\beta}$ as follow:

beta	6.5859
eta	62.9566

(5) Check: calculate sample data points for the Weibull distribution $F(t) = 1 - \exp(-(t/\eta)^{\beta})$ and plot on top of original data to verify that the fit is acceptable.

The result is shown in Figure 9.11.

Discussion

The method accepts any positive values of measure and cumulative probability. It therefore has flexibility to accommodate a variety of user requirements regarding confidence interval provided. It is not necessary that the confidence interval be symmetrical, and Figure 9.12 shows the results for different parameters from Table 9.4.

Measure t	User probability F(t)
40	0.2
50	0.5
74	0.8

Table 9.4: New estimates

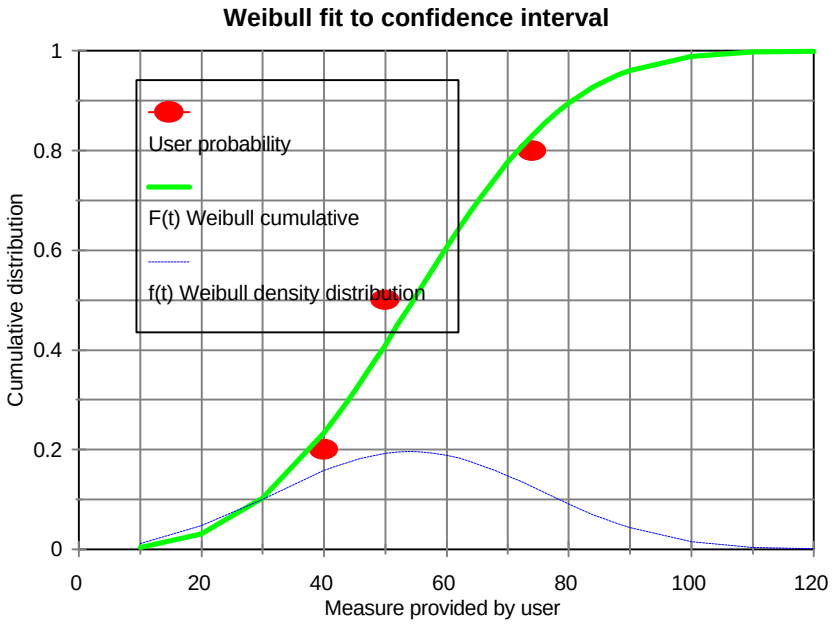


Figure 9.12: Weibull fit to different parameters.

It may be observed that changing the confidence interval broadens the density function in this case, corresponding to lower accuracy of the user's estimate.

Note also that not every discrete distribution that the user submits will be exactly fitted by the Weibull distribution, as here. This is not necessarily a major limitation, as realistically no probability distribution is going to be able to accommodate every possible user input, though some may be better than other in certain cases.

9.4.4 **Implementation**

To implement the method requires thought from the user since it requires that subjective measures be reduced to a confidence interval. A minimum of two data points are required, although the method has been illustrated above with three points.

The following code implements a system that requests three parameters from the user, and then calculates a Weibull cumulative probability distribution curve to fit. The screen is shown in Figure 9.13, and the code follows. The software has also been provided with the ability to fit a Normal curve to estimates, though that is not described here.

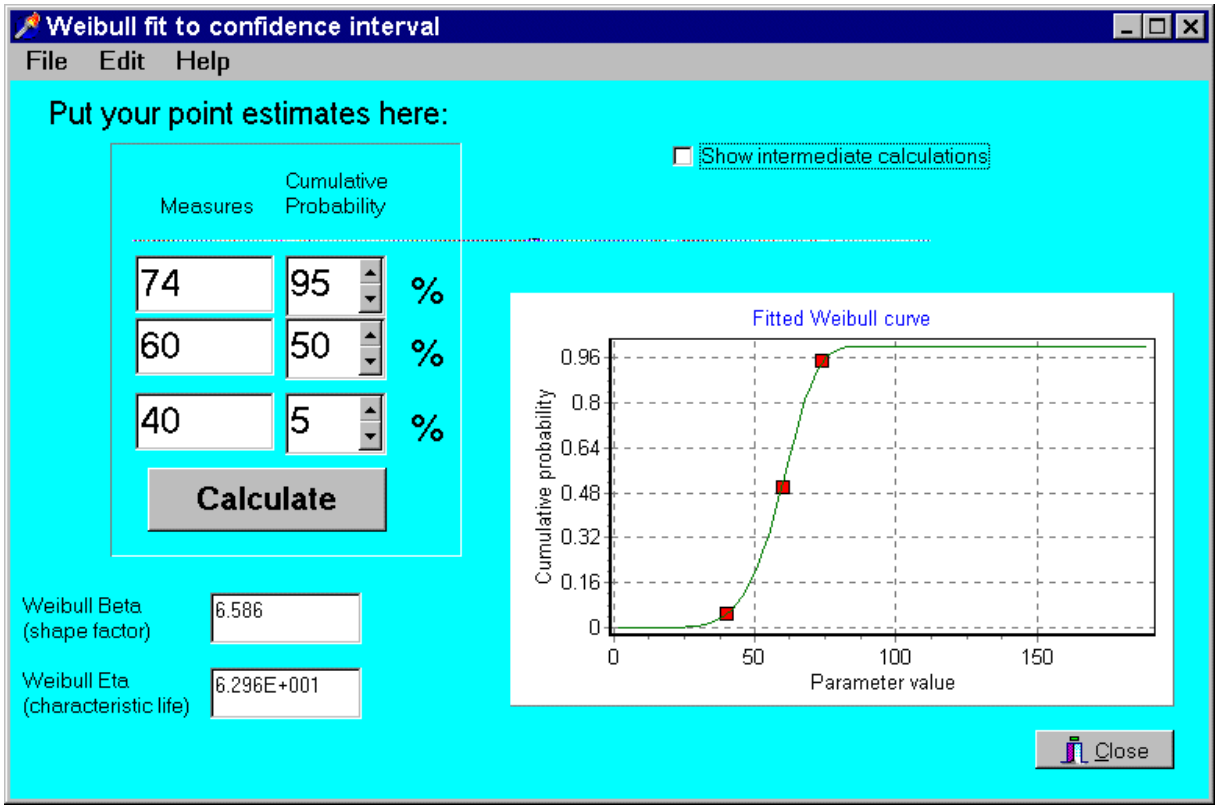


Figure 9.13: Weibull estimation software

```
{Apply Weibull transformation}
x1:= ln(T1);
x2:= ln(T2);
x3:= ln(T3);
y1:= ln(-ln(1-P1));
y2:= ln(-ln(1-P2));
y3:= ln(-ln(1-P3));

{Plot y(t) against x(t) and fit a straight line using a least squares.
Determine the slope a and the intercept b}
slopeA:= (3*(x1*y1 + x2*y2 + x3*y3) - (x1+x2+x3)*(y1+y2+y3))
        /(3*( (x1*x1)+ (x2*x2)+ (x3*x3)) - (x1+x2+x3)*(x1+x2+x3));

meanX:=    (x1+x2+x3)/3;
meanY:=    (y1+y2+y3)/3;
interceptB := meanY - slopeA*meanX;
```

```
{Calculate  beta = a and  eta = exp(-b/beta) }
beta:= slopeA;
eta:= exp(-interceptB/beta)  ;
{Define Weibull function}
Maxt:=3*eta;
for j :=1 to 100 do
begin
    time[j] := j*Maxt/100;
   Weibull[j]:=1-exp(-1*Power((time[j]/eta),beta)) ;
end;
```

9.5 **Histogram Monte Carlo algorithm**

The software developed in this project also includes a Monte Carlo algorithm so that benchmarking may be done. Many Monte Carlo tools use an algebraic relationship for the cumulative probability, and one that may be rearranged to give an explicit inverse function of time as a function of probability. Use of a histogram was found to be poorly documented, and the author was forced in 1997 to develop an algorithm for the purpose. Subsequently this was discovered to be simply a re-invention of an algorithm already described by Manno (1999, p45), so the discussion here will be brief.

For a histogram the procedure to find a time value is to first generate a random number between 0 and 1. Then the time value is identified as that histogram interval for which the cumulative probability equals the random number. This method also preserves the distribution shape, i.e. it will not over sample the tails. While the description may be brief, its implementation requires more programming. It also makes a significant computation overhead to the Monte Carlo process. There is much internal searching that must take place to find where the point lies, and this searching is more time consuming than use of an inverse function. However this is simply unavoidable for the histogram case. An extract of the source code follows:

```
//-----------------------------------------------------------
//SET UP INPUT WEIGHTED SCALES
//Determine prob x delta time (pt) points on the weighted scale}
//Done for each of the input parameters (dimensions)
//-----------------------------------------------------------
```

```
//PROCESS FIRST INPUT PROBABILITY DISTRIBUTION
//-------------------------------------------------------------
//Calculate Time at the upper and lower bounds of this interval
NumInterval[1] := High(Time1) + 1 ;
for i := 0 to Steps1-1  do
begin
  if i = 0
  then BinSize1[i] := abs(Time1[i]-Origin1)*2 //UpperLimitI :=  Time1[i] + (Time1[i]-Origin1);
  else BinSize1[i] := (abs(Time1[i] - Time1[i-1]) - 0.5*abs(BinSize1[i-1]))*2;
  //Calculate upper and lower limits for this interval. Will reuse name.
  UpperLimitI := Time1[i] + BinSize1[i]/2;
  LowerLimitI := Time1[i] - BinSize1[i]/2;
  //--------------------------------------
  //Calculate Time at the upper and lower bounds of this interval
  //pt point is previous pt point plus prob x (change in time)
  if i = 0
  then pt[1,0] := Prob1[0]*BinSize1[0] //(2*(Time1[0] - Origin1))* Prob1[0]  ;
  else pt[1,i] := pt[1,i-1] + Prob1[i]*BinSize1[i] ;
//-------------------------------------------------------------
//Repeat for other input distributions
...
...
...
{The above are set-up procedures which only have to be done once.}
//-------------------------------------------------------------
//MONTE CARLO CALCULATION LOOP
//-------------------------------------------------------------
//Create random numbers and calculate output
McOut := 0; //initialise
Randomize;  //Initialise Dephi randomisation
for k:= 1 to McRuns do
begin
  //-----------------------------------------------------
  //Create random input numbers
  for a:= 1 to Dimension do //repeat over all the inputs
  begin
    //Use random number generator "Random" to create a point (0..1),
    //and multiply by the maximum on the weighted scale.
    //Result is a random number between zero and the max weight.
    Chancept[a]:= Random*pt[a,NumInterval[a]-1];
    //Find out where this point falls in the weighted scale
    //If nothing is found then the point is below the scale
    Positionpt := 0;
    for i := 0 to NumInterval[a]-1 do
    begin
      //Check random point against the discrete points in the weighted scale
      if (Chancept[a] >= pt[a,NumInterval[a]-1]) then
      begin
        Positionpt := NumInterval[a]-1 ;
        Break;
      end
      else
      begin
        if (Chancept[a] <= pt[a,i]) //and  (Chancept[a] <= pt[a,i+1])
        then
        begin
          Positionpt := i;
          Break;
```

```
      end;
    end;
  end;
  //Determine chance value of time
  if  Positionpt = 0
  then  ChanceTime[a]:= Start[a]
  else
  begin
    if a = 1
    then  ChanceTime[a] := Time1[Positionpt]
    else
    begin
      if a = 2
      then ChanceTime[a] := Time2[Positionpt]
      else if a= 3 then ChanceTime[a] := Time3[Positionpt];
    end;
  end;
//-----------------------------------------------------
//Above has generated the chance times for each input distribution.
//Now combine with some operator
a:=0;
case Operator of
  //-----------------------------------------
  500  :   {ADDITION}
  begin
    McOut := ChanceTime[1] + ChanceTime[2];
  end;
  {Other operators not shown here}
  ...
  ...
  ...
```

This algorithm has the advantage like Latin Hypercube Sampling (LHS) of producing random time values that are faithful to the input probability distribution, though different mechanisms are used to the same end. Unlike LHS it does not require an algebraic inverse probability function, but can use a histogram of any origin. Unlike LHS it is not necessary to impose a ban on the re-use of the intervals previously used. It results in a set of samples given as the mid points of the histogram intervals, i.e. samples are not spread across the histogram interval. It is a relatively demanding computational method as it requires the initial creation of the cumulative probability array, followed by comparisons to check where on that scale the random value lies. Nonetheless it successfully accepts histograms and faithfully reproduces their shape. The algorithm has been built into the DSI software where it is available as an alternative computational method or as a benchmarking tool.

9.6 **Fuzzy arithmetic**

The fuzzy method of alpha level cuts has also been implemented in DSI. The cut method has been described above in Chapter 4, and the software implementation is not described further.

9.7 **Structure of 'prb' probability file**

The DSI software produces a probability file at each stage of calculation. This file is saved to disk (to the default directory) with 'prb' extension. The file type is comma separated variable (csv), so that it can be imported into a spreadsheet or word processor.

An example of a probability file follows:

Title,Cleaning.Cycles.prb,,,,,,,
genesis,betapert,3,5,8,500,,,
FileVersion,6,,,,,,,
Rows,50,,,,,,,
FileType,N,,,,,,,
DataOrigin,3,,,,,,,
Alarms,,,,,,,,
0,3.05,2.48395058679116E-256,2.48395058679116E-256,,,,,
1,3.15,1.44662443560954E-163,1.44662443560954E-163,,,,,
2,3.25,6.54134457346887E-122,6.54134457346887E-122,,,,,
3,3.35,1.85813447840439E-95,1.85813447840439E-95,,,,,
4,3.45,1.84291012920094E-76,1.84291012920094E-76,,,,,
etc
47,7.75,8.7157934871882E-250,1.0000000000262,,,,,
48,7.85,1.56786202779165E-314,1.0000000000262,,,,,
49,7.95,0,1.0000000000262,,,,,

The contents of the file are interpreted as follow. Note that the first line is named Line 0. This is a convenience since Delphi begins all line and array elements at zero rather than one.

Line 0: Contains title information which will be displayed in the chart in DSI

Line 1: Brief description of the genesis. In this case a betapert distribution with given parameters. If the data result from a calculation then the genesis

gives the parents and the operator, eg

Genesis,calculate,-,Rack.Spacing,Dishware.AxialHeight

Line 2: File version, which will usually be 5 or 6 in this project (DSI version 4.6). Earlier files had a different structure. DSI is backward compatible with many of the earlier file formats.

Line 3: Number of data rows in the file.

Line 4: File type, which will be N for numerical or T for Textual. This determines how the data are treated by the algorithms.

Line 5: Data origin is the start of the numerical scale. For textual files this value is empty or zero as it is not needed. However it is a crucial parameter for numerical data as it determines the bin widths for the numerical scale. In this example the origin is 3.00. The first data point (line 7) is given as 3.05, and this is the centre point. The bin width for the first (0^{th}) bin is twice the difference, namely 2x(3.05-3.00) = 0.10. The bin width is always assumed to be symmetrical, so it ranges from 3.00 to 3.10, with centre 3.05. The next bin (interval 1) has a centre point given in Line 8 as 3.15. It is already known that the previous interval stops at 3.10. This and the centre point are used to determine the half interval and therefore the upper point of the interval. Therefore interval 1 ranges from 3.10 to 3.20. It so happens that this interval has the same bin width as the previous one, and this is generally how the data are arranged. However it is possible to have bins of different width, though this is less common. In all cases the algorithm interprets the data starting at the origin and determines the bin width from the given centre points of the interval. The immediately previous bin width therefore affects the next bin width. If the origin is somehow inappropriate (either a result of manual editing or an unconditioned textual distribution that is being converted to a numerical one) then it is possible to create the error of negative bin widths. The software will detect this, and advise the user to apply one of the origin-reconditioning steps in the model.

Line 6: Alarm data are given here.

Line 7 to end:

Probability data with the first element (column 0) being an index (non-essential), then column 1 with the centre point of the time interval (here the number of wash cycles rather than time), and then column 2 with the probability for that interval (as a histogram). If there is a fourth element (column 3) then it is the cumulative probability. This is not always saved in the file as it is readily recomputed as the running total of the probabilities in column 2. For a textual file the data in column 1 are strings.

The commas strings are an essential part of the file and should not be deleted as they delineate blank data. The probability data (column 2) are histogram data in that they give the total probability for that bin. If the true density is required then that is given by the probability divided by the bin width. The chart inspector form in DSI can perform these calculations and show true density (see the Setup tab on that form), but the data saved to file is always histogram data regardless of what the chart displays.

9.8 **Validation of DSI software**

Software integrity is determined through *testing*, *verification* and *validation*. Deutsch (1982) defines verification as the ‘activity that assures that the results of successive steps in the software development cycle correctly embrace the intentions of the previous step’ and validation to ‘ensure that the software end item product functions and contains the features prescribed by its requirements specification’ (p xii). Alternatively, Osterweil (1984) sees testing as the ‘process of looking for errors’ and verification as the ‘process of demonstrating the absence of errors’ (p85).

It seems preferable to clarify terminology rather than get tangled in it, so this discussion uses *valid* to mean *defensible* (cf *validus* strong), *verify* as the *proving of truth* (cf *verus* true), and *integrity* in a general sense for software quality.

Simulation systems are only an approximation of the actual system, regardless of how much effort is put into the simulation. The validity of a model is established by comparing its output with that of the real system (if available). Furthermore, increases in the validity of a model may come at the cost of extensive data collection, which could decrease the cost-effectiveness of the process (Law and Kelton, 1991). Those authors also observe that “simulation models are generally better at comparing alternatives than at determining absolute answers” (p307). They also suggest that *validation* can occur by building a model of an existing system before modelling the unknown system. Law and Kelton believe that though statistical tests exist for validation purposes, these are not directly applicable due to the complexities of non-stationary distributions and autocorrelated data. In addition they note that it may be possible to use a *Turing Test* to validate a model. A simulation model passes the Turing Test if an expert cannot differentiate between the results of the real system and the simulation. Field tests are another option in some cases, involving a prototype being tested under limited conditions. If the model and the field test results agree, then the other outputs of the model (for conditions not covered by the field tests) may be acceptable.

Checking against specification

Part of the need for integrity is that most large software projects involve many people, and the integrity of the many efforts needs to be managed. In particular there is a strong need to ensure that the function required by the customer has been delivered by the programmer. There is the practical difficulty that the functional *specification* cannot usually be completed before programming, and instead must be composed concurrently with the software development (Gourlay, 1984) but even so the delivered function can eventually be checked against the needs. Checking against the specification was not an issue in this project since there was only one programmer who was also the intellectual driver, and thus there was an intimate and constant link between intent and software code. Other aspects to software integrity usually include budget time, cost and operating environment (Deutsch, 1982) but these were of limited relevance.

Checking internal integrity

Validating the internal steps of a program is an important objective. How can we be confident that there are no significant errors in the structure of the program or its variables? One approach is to treat the program as a theorem and seek to prove it formally using *symbolic execution*. However culmination of formal verification is rarely reached since there are residual uncertainties and the symbolic representation may simplify the real execution (Osterweil, 1984).

Since formal verification is often impractical and uneconomic, Deutsch (1982) (p11) instead recommends developing confidence in the program by using it on test cases. Proper operation of the software is deemed to be provided when the number and severity of errors falls below a threshold, although Deutsch acknowledges that 'proper operation' can be difficult to define. However he reports that to demonstrate successful operation through every possible combination of paths through the code, or all possible combinations of inputs, is utterly futile as billions of years of computing time could easily be required. Instead a small number of test cases may be used to adequately test the code. The systematic approach to testing would then be to decompose the overall intent into smaller requirements, and then design, implement and document a test for each. This testing may be done in an incremental manner and concurrently with software development. However automatic test tools are usually required to support this process (Deutsch, 1982).

Several methods were used to ensure integrity of the DSI software. Use was made of *probes* (Osterweil, 1984). Probes are fragments of code inserted where errors might be expected. During execution these probes operate and test whether the results are as anticipated. Osterweil (1984) motivates for an extensive implementation of probes, and the successive removal of them once each has been found by formal symbolic methods to be unnecessary. This is a rigorous and time consuming approach and instead for DSI the author used a less formal approach. The probes were implemented as new algorithms were being written, and subsequently removed when

it was clear that results were as expected.[144] It is desirable to remove unneeded probes as they add to the size of the compiled file. There are still many reporting probes in place and these can be activated by selecting appropriate checkboxes.[145]

The integrity of the DSI code is further enhanced by quality control features provided in the development language. It is relevant to note that the DSI software was developed in *Delphi*, which is a high level programming language with extensive debugging tools and interface checking. Delphi produces a compiled executable, and during the compilation process it imposes semantic and structural integrity: the application will not compile at all until such errors are resolved. In contrast other development software like Visual Basic (VB) provides on-the-fly compilation so primarily only those routines that are used are scrutinised by the compiler. The Delphi compiler analyses (among others) semantic correctness, the completeness of the code, the procedure definitions (including the consistency of calls to procedures and the types of variables passed), the presence of undeclared variables, and variables that are used inconsistently compared to their definition.[146] This analysis occurs at compile time and the developer cannot avoid it. Each time a build of the software is constructed, Delphi checks its basic integrity and halts compilation if there are errors. Delphi also identifies the location and type of error, and this significantly assists the human programmer.

Another important feature of Delphi is that during development the user's application runs inside a protected space. The Delphi debugger monitors the application, the line being executed and the states of all variables. If an error occurs then the Delphi

---

[144]The probes in DSI are 'ShowMessage' functions that show a message at run time depending on the value of a preceding conditional statement.

[145]The checkboxes may be found at Options/Options../General and then check 'Show Details during working'. For yet more detail reporting activate checkbox 'BoxExtraDetailsRequired' if it shows on your version of the software. Checking both boxes will also activate code that saves all the intermediate combinatorial probabilistic computation results to disk as file AJ.txt.

[146]Delphi is restrictive about the typecasts it permits. For example it will not permit an integer to be used as a real (or vice versa). This imposes integrity on the code because variables cannot be used inconsistently. Instead the developer has to actively provide additional variables and make a conscious decision to copy and convert the variable into the new type. Other programming languages such as C++ are less restrictive about typecasts.

debugger automatically intercepts it, identifies the type of error and the line where it occurred. The user can inspect the values of all variables. Thus the error handling is very much more informative and therefore powerful than can be provided by the Windows error messages. In this project it was found to be particularly valuable in identifying arrays that were out of bounds. This type of error involves addressing an array index outside the previously defined lower and upper bounds (eg seeking data in element 102 of an array that runs from index 0 to 100).

A large number of arrays are used in DSI, and it was necessary to ensure that the appropriate start and end points were used on all the many array operations. It is usually not possible for a compiler to identify this type of array error as it typically occurs when a loop indexes too far, which occurs at run time. However the run-time debugger successfully catches these errors and identifies the source code line where the error was observed. Thereafter the developer must *debug* the error, by tracing it back to its source.[147] The author would be reasonably confident that no significant out-of-bound errors exist on the main computational paths through DSI, since this type of error was monitored by Delphi during the entire development process.

Delphi provides the ability to define *dynamic arrays*.[148] For the user the advantage is that small probabilistic computation models take only a small amount of memory to execute and can therefore be practical on modest computers, but models can easily be scaled up to finer resolution (and therefore bigger internal arrays) as increased computation power becomes available. For example, DSI might typically be run with each input distribution modelled by 50 points. This gives reasonable quality results in reasonable computation time. However when computational power becomes

---

[147] If there are out of bounds errors in an application then MS Windows could let the application access whatever memory was adjacent to the array, corrupting the process or the data. Errors with array indices are therefore disastrous. Furthermore if the data were outside of the memory space of the application (i.e. adjacent data were from another windows application) then Windows would declare a 'general protection fault' and shut down the application.

[148] By comparison earlier software required that the size of an array be fully defined in the code. This imposes the constraint that the code has to be edited if longer or shorter arrays are required. Also, efficiency is not always high: for reserve capacity a developer might provide larger arrays than necessary, and this consumes memory. With dynamic arrays the size of the array is determined at run time. This means that an array need only be as big as is required at that time, and next time that code executes it could be a different size. DSI makes extensive use of dynamic arrays.

available then the model could be run with inputs of say 1000 points, without having to edit the source code. However for the software developer dynamic arrays are more complex than static arrays. Their size has to be defined using run time statements, and they have to be freed at the end. Failure to free them causes *memory leaks*, which though not critical do consume memory and cause the application to slow. The dynamic arrays are owned (in a Windows sense) by the application and therefore closing the application will free up any memory that may inadvertently be tied up in arrays. The author would be reasonably confident that dynamic arrays have all been freed after they are used. However if execution terminates unexpectedly then it is possible that the code to free the arrays might never be encountered. Therefore in critical areas this code has been placed in protected blocks. These Delphi constructions (of the form '*try .. finally..end*') ensure that regardless of how the software terminates (other than a power failure) the code in the 'finally' block is always executed.

Delphi also provides that *break points* may be inserted into the code. When the application is run inside the Delphi development environment then execution runs until it reaches a breakpoint, and then pauses and transfers focus to the appropriate point in the source code. The developer can then inspect the values of all variables. Values of variables appear as flyby hints.

Once out of the Delphi development environment the application does not have the debugger monitoring it. Thus a deployed application such as DSI runs faster (there is no monitoring overhead) but also there is no further error interpretation as MS Windows handles the errors directly.

The Delphi compiler and debugger were most useful in intercepting and identifying various software errors. They have contributed significantly to the integrity of the DSI software. Since the compiler is prescriptive it has ensured that there are no errors of syntax or undeclared variables or type abuse, and that all procedure calls are consistent. In addition the run time debugger has likely caught all array out-of-bounds errors on the main computational paths. The author is confident that the DSI software

contains no significant errors in the structure of the program or its variables (where significance is defined as compromising the intent expressed in the project thesis).

Combined with the mathematical validation provided in Chapter 4, the DSI method has thus been validated by several methods. Validation refers to it being defendable rather than proved. It could be a project in itself raising the validation to a higher level of confidence, or attempting a formal symbolic proof, but this would not add value to the current project or its central thesis.

9.9 **Conclusions**

The DSI software has been validated by reference to the software engineering processes used in code generation. The mathematical algorithms have been validated by demonstrating that they produce results that are in agreement with the algebra of random variables.

Chapter 10

Probabilistic approach to life cycle producer cost

The life cycle costs of a dishwasher development are modelled quantitatively with the Design for System Integrity software.

10.1 **Introduction**

Life cycle costs are the totality of all the costs that affect the product over its life. The life cycle of a product includes the development, production and eventual decommissioning costs of the product, and the life cycle analysis concerns the costs and incomes over this period.

The costs may be divided into two broad groups: producer's costs and consumer's costs, though this chapter addresses only the producer's viewpoint. This cost is made up of numerous elements. The Producer in this context includes the manufacturer, assembler, and seller of the machine. These functions may be distributed between multiple physical organisations, but for convenience are here modelled as one. The Producer's viewpoint looks at the long term profitability to the manufacturer of the product.

Existing approaches to life cycle cost

The conventional approach to *life cycle cost* is first to identify the cost elements, then to estimate the monetary cost. Since the life cycle of a product includes the development, production and eventual decommissioning costs of the product, many costs will be estimates of future spending or income. It is usually necessary to refer those values back to today's value. The mechanism for this is *present value*.[149]

---

[149] Present value is given by:

$$PV = \frac{x}{(1 + i)^n}$$

Where
n number of years into the present
x monetary value of cost or income in year n
i interest rate (per annum)
The *net present value* (*NPV*) is then the sum of the individual costs and incomes. Each of these may occur in a different year and will therefore have their own value for n in the above equation.

The application of risk analysis during product development is widely promoted in the literature for example by Garcia (1994), Ridgman (1996), Mar (1991), Henriksen (1997). The life-cycle costing applications also appear, such as Marshall (1990) who explores the financial evaluation of building investments, and Wen and Kang (1996) who studied life cycle cost of buildings.

Conventional modelling work, including life-cycle costing, is generally based on single point values, which are propagated through the simulation. Such approaches give no indication of the probability of the simulated outcomes, and there is no way to check the risk of the outcome. The integrity of the final estimate can therefore not be established.

Sensitivity analysis is sometimes used to provide a check of this. Single parameter sensitivity analysis provides results that are easy to interpret, eg in the form of a *tornado diagram*. This can be used to determine which parameter has the greatest influence in the outcome. However this imposes the assumption that only this parameter is variable. The reality is that in most cases many other parameters are also variable at the same time. Methods exist for *multi-parameter sensitivity analysis*, in which case the problem becomes one of determining which combination of parameters is critical to the outcome. Sensitivity analysis might show which parameters are more critical in determining the output, but the method is limited in that probability of a parameter is left undefined: the method is deterministic rather than probabilistic.

The barrier to probabilistic approaches is that there is no universal mathematically explicit solution to the problem of combining probability distributions, other than the addition or subtraction of special distributions (eg Normal). Sarper (1994) discusses the inclusion of risk into capital investment decisions. However Sarper uses the normal distribution as an approximation to uniformly distributed random variables, an approximation that is arguably crude.

A more powerful method, and one that provides full quantitative probabilistic computation capability, is *Monte Carlo* analysis[150]. This has been used in a variety of domains, including financial modelling (Duckworth et al, 1998; Zhang et al, 1992; Kleijnen, 1995)[151]. The combination of Monte Carlo analysis and life cycle costing has been used in some cases. For example the financial investment decision regarding the economic viability of domestic solar energy systems is modelled this way by Walley et al (1990). They calculate consumer's net profit based on factors such as lifetime of the equipment, future cost of alternative energy, and available finance. They used a probabilistic model. Shu et al (1996) apply probabilistic methods to analyse life-cycle cost of fastening. They apply a reliability model to fastening systems to determine re-manufacture cost. They proceed to combine this with other related costs. They apply *genetic algorithms* to optimise combined life-cycle cost.

This chapter describes a model for dishwasher life cycle cost, using the Design for System Integrity (DSI) probabilistic computation tool.

10.2 **Explanation of the DSI methodology**

The DSI methodology can process both quantitative and qualitative relationships, though this paper only uses the quantitative analysis. Quantitative analysis in this context means that the variables being considered are numerical in nature, and represented by probability distributions. Furthermore, the relationships between variables are precisely defined by mathematical functions, i.e. there is no uncertainty of analysis. By accommodating a probability distribution for each variable, the DSI method permits process variability to be modelled. More importantly, it permits the

[150]This method takes random samples from the input distributions and determines the outcome. The method can determine the probability distribution for the outputs of a model if sufficient random samples are taken. It is able to accommodate any input distributions.

[151]Croll (1995) feels that risk analysis methods such as Monte Carlo simulation will spread in usage in engineering projects. Croll believed that more powerful risk analysis software tools are necessary in order to make the methods "more accessible to the general manager". As Croll points out, one of the major benefits of risk analysis, despite the lack of software tools, is that it "admits to the modelling process ...the existence of uncertainty".

effect of process variability to be propagated through the model. In this way the process variability of each and every parameter affects the final outcome. As a management tool, the method may be used in the early stages of product development, such as in feasibility studies, since it accommodates the high levels of variability inherent in this stage.

The DSI method combines distributions with a combinatorial mathematical function. It avoids Monte Carlo simulation altogether. Instead it takes the two input probability distributions, converts them to discrete distributions, and then determines the resulting output distribution piecemeal. The pieces are then combined to create the output distribution. There is no limit on the input distributions, since any distribution may be converted into a discrete one.

10.3 **Model of Life Cycle costs**

Identifying the cost horizon

The domain of interest in this application is to determine the life cycle costs for a manufactured product, namely a dishwasher. One of the immediately apparent costs is the Manufacturing cost. This can be relatively tractable since costs of raw materials, parts, labour and infrastructure are often easily identified in accounting systems, at least for devices that are in production. However even these costs are difficult to identify in the early design stages, especially for products that require a significant departure from existing manufacturing capability.

Other costs are very much more tenuous. For example the capital cost to commission a plant can be difficult to estimate accurately. Even more difficult is the decommissioning cost at the end of the operational life of the plant, especially as responsibilities and social expectations may have changed in the intervening years.

This model looks at producer's cost as the profit on the product, where that profit is adjusted for present value. The model developed in this project for producer's cost is

shown in Figure 10.1 below. This is a typical rather than an exhaustive list of cost components, and the data are simply representative.[152]

Producer profit

The Producer profit is the Nett profit less the Venture costs. Each of these is further broken down. The model takes into account product development cost, plant commissioning and de-commissioning, overheads, production cost, warranty failures, liability costs, and sales volume.

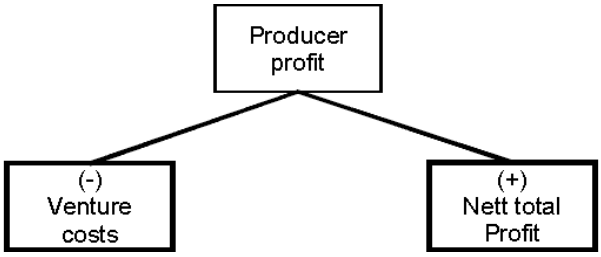


Figure 10.1: Top level model for producer profit is nett total profit less venture costs.

Venture costs

The Venture costs consist of Product Liability, Product Development, and Plant Setup. Figure 10.2 shows how these have been further modelled. In particular the model takes into account the staff costs during product development, and the project duration. All the parameters in this model are modelled with probabilistic values. In the early stages of product development the data are sparse, and therefore we use confidence intervals to model the probability values. Continuous probability Weibull distributions are then fitted to these confidence intervals.

[152] Manufacturers are understandably reluctant to transfer their financial costings to the public domain.

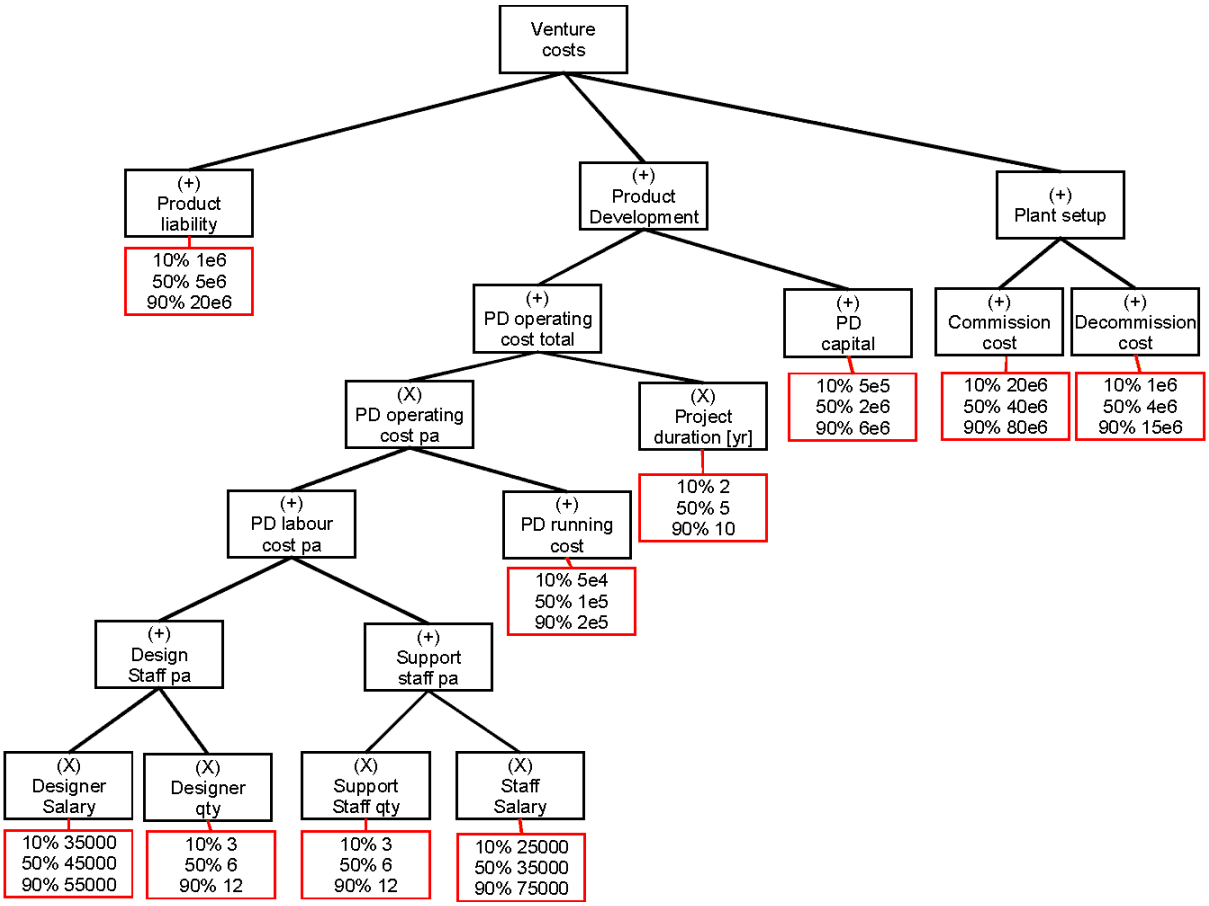


Figure 10.2: Breakdown of Venture costs into hierarchical cost elements. Computation begins at the bottom of the tree and progresses upwards. Confidence limits are given for all the primary assertions, and these are shown in boxes.

The DSI method is used to perform probabilistic reasoning according to this chart. The result, for Venture Costs, is shown in Figure 10. 3 as a probability density. The median value is obtained from the cumulative chart (not illustrated) as $63.6e6.

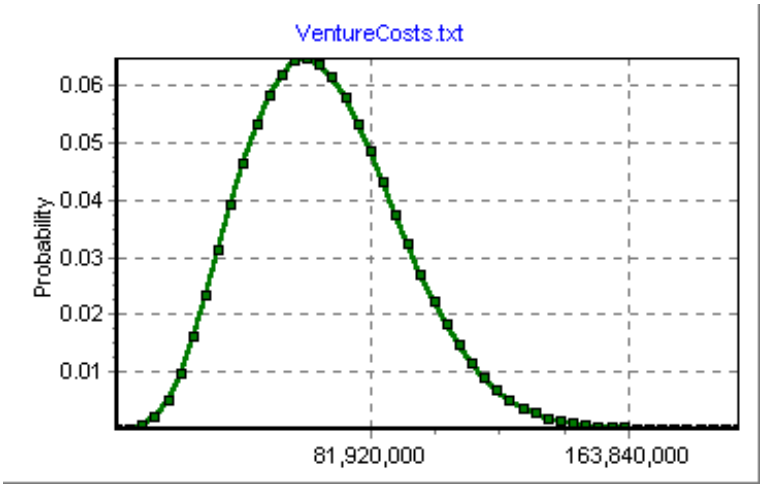


Figure 10.3: Venture cost as a probability density distribution, produced by propagating the assertions though the model shown in the previous figure.

Nett total profit

The Nett total profit is defined here as the annual profit (sales less expenses) accumulated over the product lifetime. The model is shown in Figure 10.4. Again the input parameters are modelled as probability distributions, using confidence intervals.

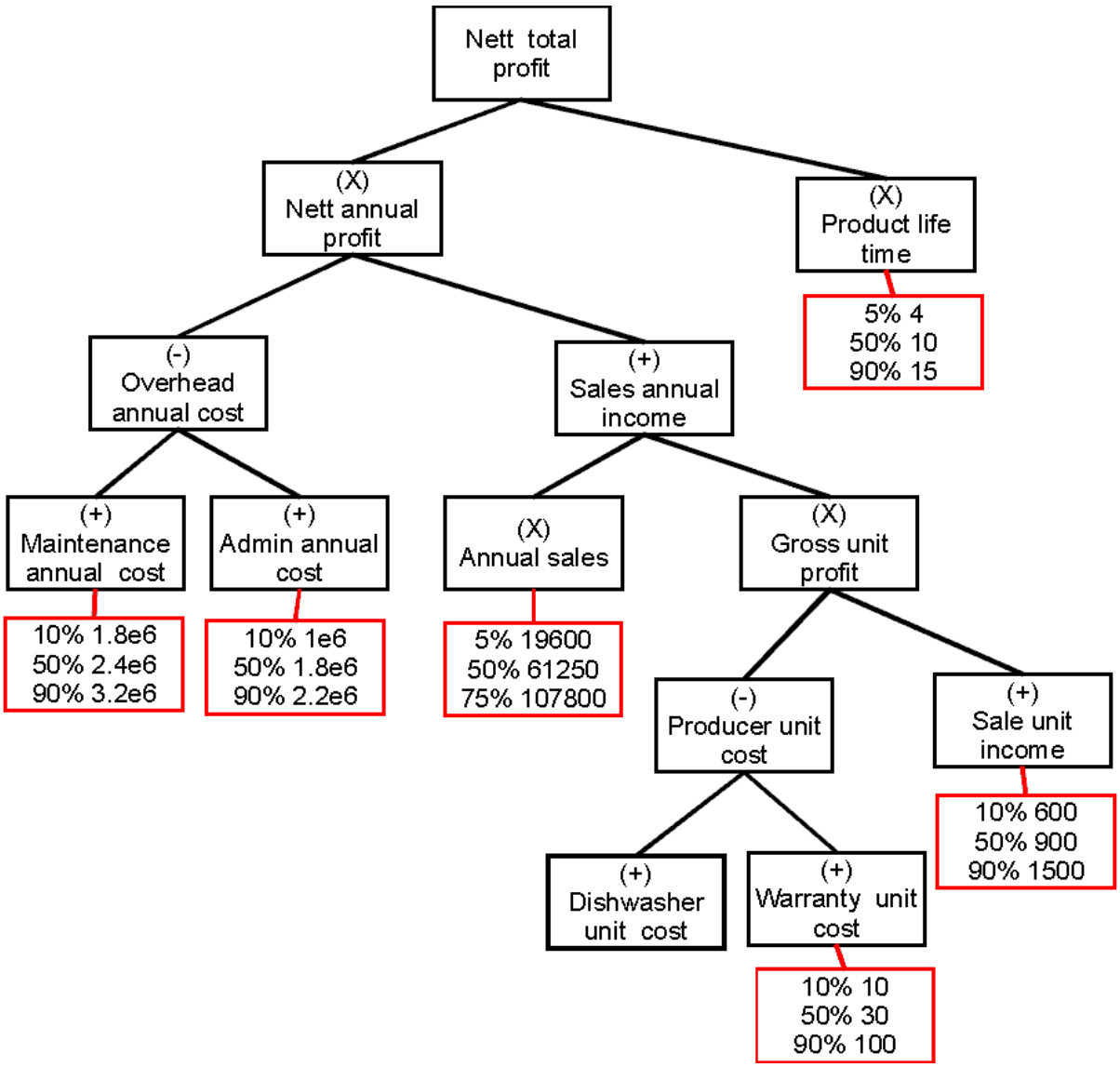


Figure 10.4: Nett total profit is computed as a sequence of operations on asserted inputs. This tree shows the computational structure, and the confidence estimates used for the assertions.

In this model *Dishwasher unit cost* is modelled as the sum of the individual device costs and the assembly cost, as per Figure 10.5.

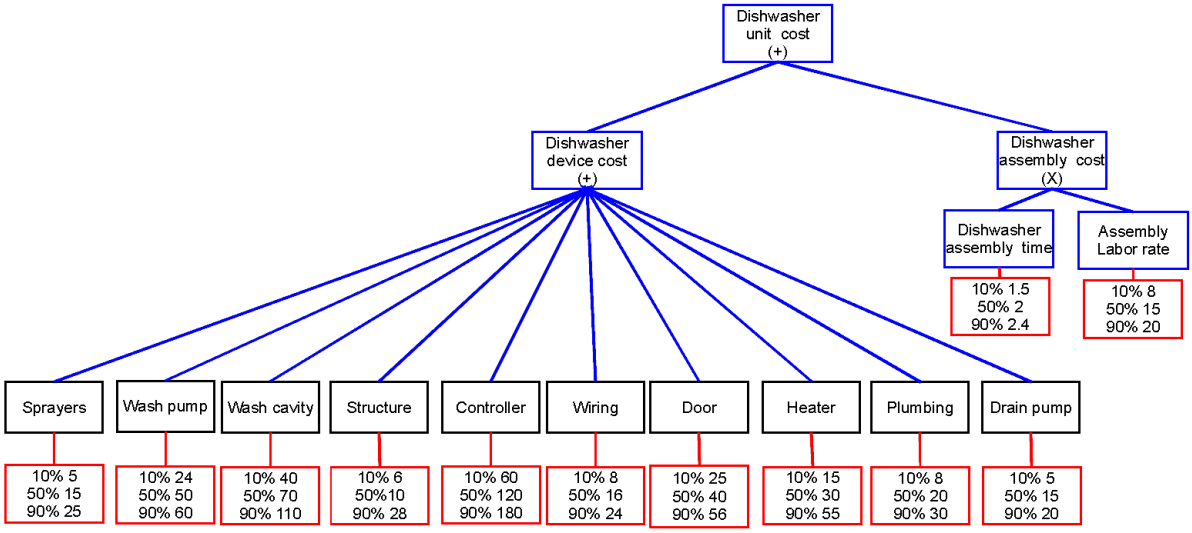


Figure 10.5: The cost of a dishwasher in terms of the component subsystems is modelled by this graph.

The model gives the Dishwasher Unit cost as a probability distribution. This is propagated through into the rest of the model, where it is combined with probability distributions for the other parameters. The result of Nett Total Profit is then determined, and its probability density distribution is shown in Figure 10.6.

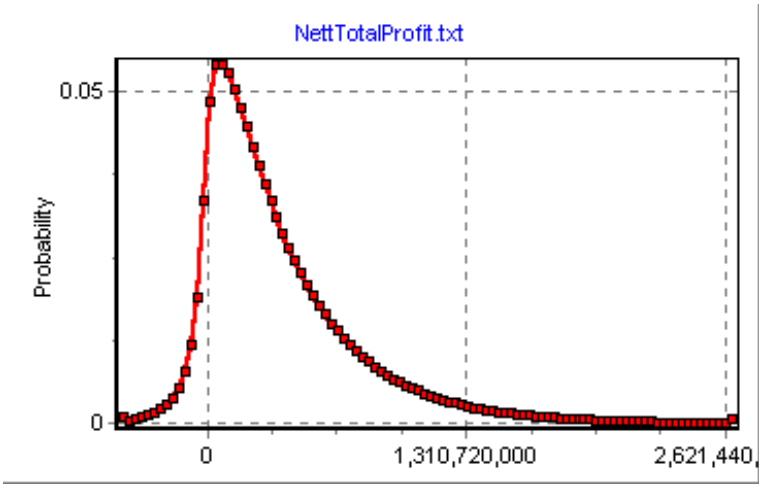


Figure 10.6: Probability density distribution for Nett total profit after simulation.

10.4 **Results of Life Cycle cost analysis**

Intrinsic to the methodology is the use of probability distributions to model the parameters. The final step in the analysis is to subtract the Venture Costs from the Nett Total Profit, to obtain the Producer Profit. The top result is the Producer Profit, and this result too is in terms of a probability distribution. This is illustrated in Figure 10.7. The charts represent probability densities.

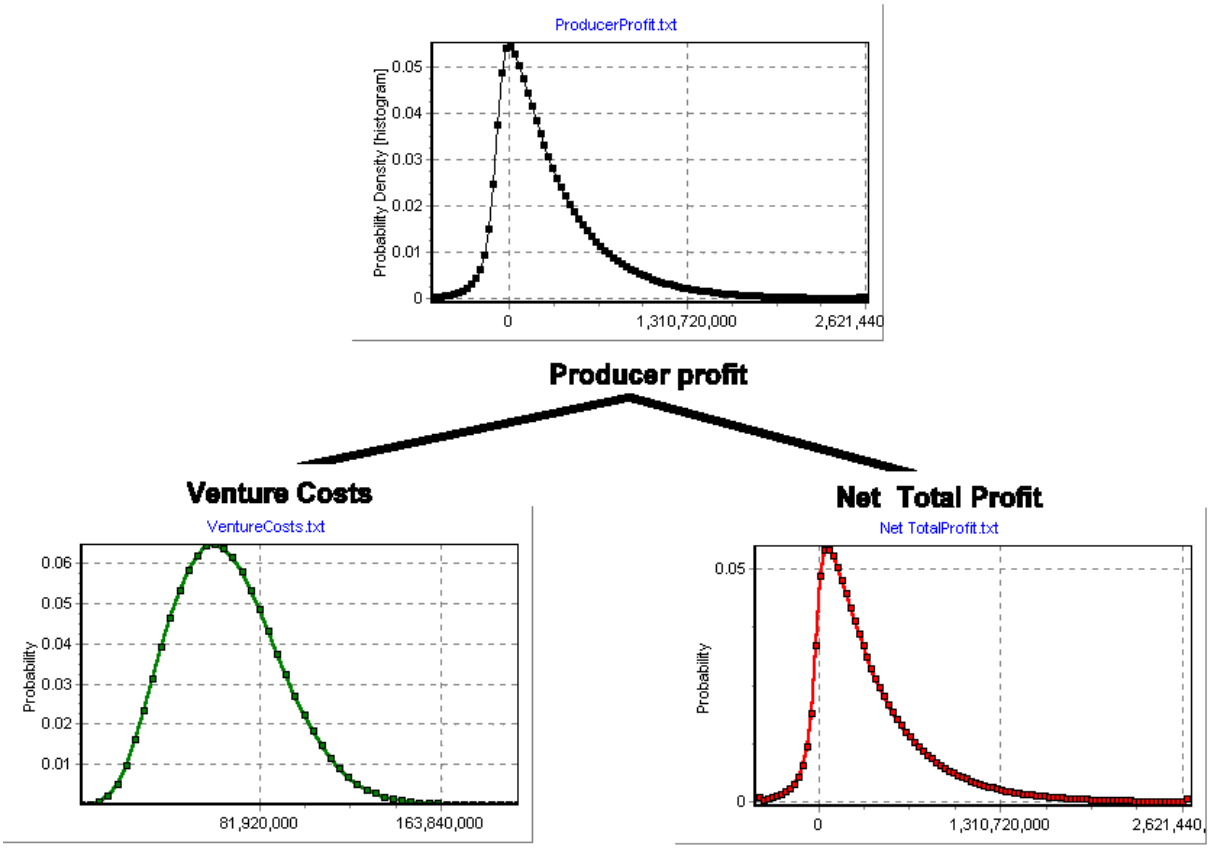


Figure 10.7: Producer profit is the nett total profit less the venture costs. This figure illustrates the probability distributions involved in the inputs, and the resulting probability distribution.

The density chart for Producer Profit shows several features:

- There is the possibility of a large positive result (large profit), though the upper extremes have low probability of occurring. A large positive tail exists.

- A smaller negative tail exists, being the region giving a result below zero. This characteristic is primarily inherited from the Nett Total Profit, where it comes about primarily due to high Overhead Annual Cost and low Sales Annual Income.
- The mode is the peak near zero, indicating that the single most commonly occurring outcome is expected to be around zero profit.

While the density distribution is readily interpreted, the cumulative distribution is more valuable for decision making. The DSI method readily produces a cumulative distribution, as it is the area under the density curve from negative infinity to the point of interest. The cumulative distribution for Producer Profit is given in Figure 10.8. Important statistics may be read from the cumulative distribution. The median profit is $169M, which should be interpreted as present value over the full life cycle. The median is the outcome that could be expected 50% of the time. This is a better indicator of project attractiveness than the mode. The density is not a symmetrical distribution and therefore in principle the mode, mean and median will not be the same.

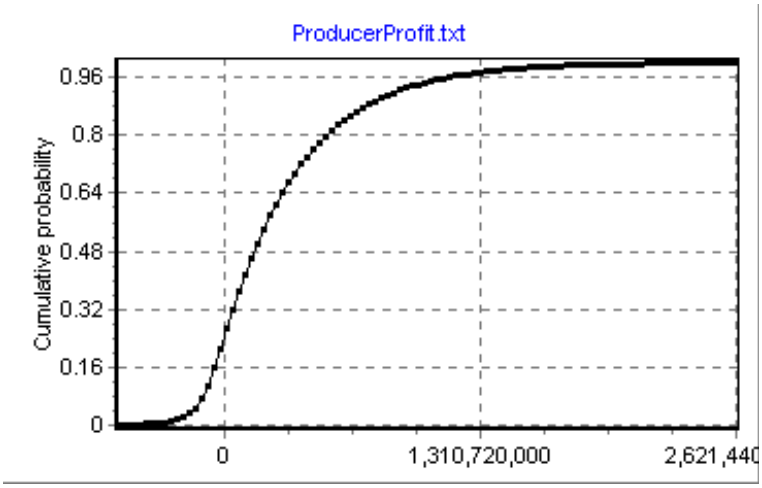


Figure 10.8: Producer profit, shown as a cumulative probability distribution.

The cumulative distribution also provides a tool to determine the likelihood of various pessimistic and optimistic outcomes. For example, the probability of a negative return is found to be 30%. Likewise the probability of a profit up to and including say $1310.7M is 0.97277, or more usefully, the probability of a profit greater than this amount is 1-0.97277 = 0.02723 = 2.7%.

In this way the DSI method produces a distribution for each parameter in the financial model. This distribution may be used to manage the project, particularly to make decisions regarding financial viability and risks. Unlike the artificial intelligence and *expert systems* that seek to make a decision, the DSI method does not make decisions itself. The method is an analytical assessment tool that supports the human decision making process by providing a mechanism to measure and process risk. It is up to the human manager to make decisions, based on his own tolerance or aversion to risk.

10.5 **Future work**

It is anticipated that the model developed here could be applicable to the analysis of consumer's cost too. The graph shown in Figure 10.9 might apply. From the viewpoint of the consumer, the parameter that dominates in many markets is the *sale value* (purchase price) of the machine. Other costs are *operating costs* (which depend on water and electrical consumption and the ownership time), *repair costs* (depend on length of ownership and the reliability after warranty), and *disposal costs* (if any). There is a link between the consumer's costs and producer's costs in that the *sale value* is an element common to both. This link could be honoured by the DSI system.

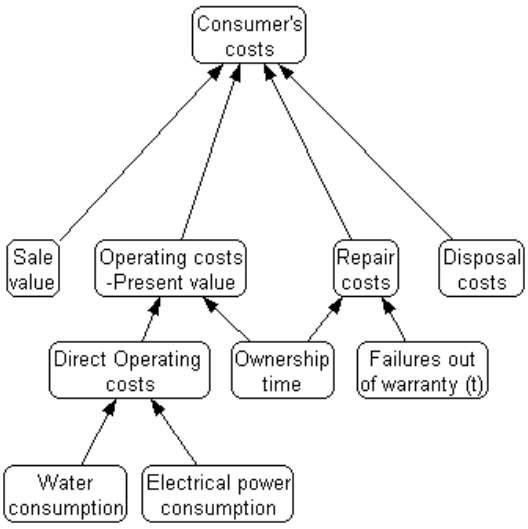


Figure 10.9: Possible model of Consumer's costs

10.6 **Conclusions**

Managing new product development ventures is an inherently difficult task. The primary financial decision is typically whether the proposed project has adequate long term viability. The decision is complicated by the lack of comparative data, especially when novel products are being proposed or manufacturing process depart from existing technologies. Furthermore the financial estimates fed through to decision makers may easily be unduly affected by beliefs held by project proponents (or opponents) at lower levels. For example, it is easy to contaminate many decision processes by supplying optimistic estimates that inflate the income streams and down play the costs. Often such estimates in themselves may be accurate, but their combination may not. That is, high income streams may be possible, but not necessarily combined with low costs. In probabilistic terms, the proponent has selected estimates from extreme values of the distributions, and the probability of two such extreme events occurring is much less than the probability of either event on its own.

The DSI method accommodates these management difficulties by providing probabilistic computation for financial models. By using probability distributions rather than single-value parameters, the manager is presented with outcome results as a probability distribution. The extremely optimistic and pessimistic outcomes are shown in the distribution. Importantly, their probability of occurrence is also provided, so such extremes can be seen in context. Moreover, since the method requires input parameters to be given as probability distributions, it imposes a probabilistic reasoning culture on those who provide the data. It requires that the person making the estimates provide for the spread in the parameter, and thereby expose their belief about that cost or income parameter. This provides a check against unreasonably optimistic or pessimistic estimates, while still allowing staff to express their beliefs. Accountability may also potentially be improved in that actual outcomes may be checked against original estimates and the necessary lessons learned.

The DSI method allows input parameters to be expressed as full probability distributions if that level of detail is available. In early feasibility studies the data are typically more sparse, and so another method is provided in the form of confidence intervals. These require the user to provide at least three estimates. These might typically be the best, average, and worst cases, which are concepts that are generally well understood.

The DSI method has been applied to predicting the producer profit for a new dishwasher development. The model has incorporated venture costs (typically those of product development and plant setup) and the income from sales over the life of the product. The financial estimates used in this study are only representative.

The key advantage of a probabilistic computation approach to life cycle costs is that the decision maker is presented with both the profit figure and the risks therein. In the past this type of simulation could only be obtained with the Monte Carlo method, and it has here been demonstrated that the DSI method is also able to provide the necessary computational framework for this kind of simulation.

Chapter 11

Simulating Wash Performance

This chapter describes the application of the Design for System Integrity method to simulating the wash performance of domestic dishwashers. The simulation combines both qualitative and quantitative parameters, and addresses the large inherent uncertainty in the parameters at the early design stages.

11.1 **Viewpoints of design**

Design is a complex task of simultaneously satisfying requirements in multiple viewpoints. A design has integrity if it meets the requirements in various viewpoints. For engineering product design the primary viewpoint of design integrity is that of performance. This is often the critical viewpoint for functional success of the product. Other viewpoints may also need to be satisfied, such as reliability, manufacturability, styling, and cost. In this chapter *performance* refers to the behaviour of the machine, in particular the measure of how well the machine meets the purpose for which it was created (the *design intent*). There can be more than one performance criterion for a machine. The difficulty is that many performance criteria are qualitative,[153] and the relationships weakly understood.

11.2 **Dishwasher performance**

The primary reason for a customer to purchase a dishwasher is to clean dishes, and therefore the main measure of dishwasher function is *wash performance*. This is typically measured as the amount of soil left on dishes after washing. The test varies according to country and test organisation. A dishwasher will not necessarily perform consistently across all these tests, and possibly a ranking of competitive products will also not necessarily be the same by all tests. Repeatability of results even within one test method can sometimes be difficult to achieve.[154]

[153]The interested reader is referred to Chapter 6 for a description of how the terms quantitative and qualitative are used in this work.

[154]Two other performance metrics for dishwashers are *energy usage*, and *water consumption*, though not the topic of this paper. Quantifying energy and water consumption is relatively simple and reproducible compared with wash performance.

11.3 **ANSI/AHAM wash index**

The ANSI/AHAM standard (DW-1-1992) describes test procedures for determining wash performance of dishwashers. It gives a "*washing index*" rating between zero (dirty) and 100 (clean). The procedure has no underlying wash theory. Instead it relies heavily on weighting factors of unknown etiology, as the following brief description illustrates.

In the test a load of dishware is soiled with various foods, left to dry for two hours and then washed on a normal cycle[155]. Afterwards each item is visually examined for soil, and each defect found attracts demerit points. The test is weighted against particle diameter, as Table 11.1 shows.

Particle size	Demerit points per particle
< 3.2 mm	1
3.2 to 6.4 mm	3
6.4 to 9.5 mm	7
> 9.5 mm	9

Table 11.1: ANSI/AHAM demerit points depend on the particle size.

Each dishware item can get a total of nine demerit points. Next the number of items n_i with i demerit points is counted (where i = 1..9). The percentage of scores 1...9 relative to the total number of dishware items (N) are calculated as s1...s9, i.e. s1 = Qty(n_1)/N. The scores are added together using a weighting method to give the wash index:

[155]The ANSI/AHAM test defines:

- Ten place settings, where a standard place setting is defined in terms of dishware, glasses and flatware (knives, forks, spoons, and cooking implements), and serving ware.
- conditioning of the test load by pre-wash.
- Water supply temperature (cold 10°C, hot 60-75°C), pressure (32.5-37.5 psi), hardness (0-85 ppm).
- Soil of cooked oatmeal and milk, cooked mashed potatoes (with margarine, milk and salt), half cooked egg yolk, cooked ground beef and tomato paste, creamed corn, coffee, coffee grounds, peanut butter, preserve (jam), tomato juice.

$$Wash\ Index = 100 - \frac{1}{8}[Qty(s_1) + 2Qty(s_2) + 4(Qty(s_4) + Qty(s_5) + Qty(s_6)) + 6(Qty(s_7) + Qty(s_8)) + 8Qty(s_9)]$$

Equation 11.1: ANSI/AHAM wash index

This index is calculated separately for dishware, glasses, and flatware. Then an overall wash index is determined as the weighted average of the three dishware categories, where the weights are the number of items in each category. An example of the method with data is shown in Appendix 2.3.

Comment

The ANSI/AHAM standard makes no attempt to justify the particle scoring and weighting methods, neither has clarifying literature been found in the public domain. Possibly the methods and weights are simply arbitrary and convenient. It is curious that it uses a weighted average on the frequencies, when averaging the raw scores would be much simpler, but the author surmises that perhaps ANSI/AHAM selected the weighted average to dampen the variation between tests. The ANSI/AHAM test produces a wash score as a percentage and therefore has a quantitative outcome. If the underlying method is arbitrary then that will affect the outcome, and indeed a dishwasher will sometimes score very differently on other wash tests. The results from any wash test that produce a single percentage imply a precision that is not necessarily there.

11.4 **Development of a model for Wash performance**

The critical questions are: what parameters in the machine affect wash performance, how might they be related into a systematic model to predict wash performance, and how could such a model be applied where information is sparse (eg at early design)? This paper proposes a model to address these questions.

11.4.1 **Strategy**

The ideal approach might have been to develop a quantitative wash model, so that the arbitrariness in wash scores could be eliminated. To do this would require an understanding of the underlying mechanics and the development of a mathematical model. The model could then be interrogated to perform optimisations and what-if type studies. Alternatively, but less precisely, if the relationships could be expressed as logical rules then an *artificial intelligence* method such as an *expert system* could be developed.

Unfortunately the public literature on wash performance is almost non-existent. The principles of wash action are undocumented, and no models of system performance exist. Current understanding of wash performance is therefore only qualitative expert opinion.

Consequently the method used here was to base the model on expert opinion where necessary, accommodating quantitative information where available. The Design for System Integrity (DSI) methodology and its software implementation were specifically developed to process the complexity of qualitative relationships alongside quantitative relationships.

11.4.2 **Model development**

A DSI model consists of a sequence of qualitative and quantitative calculations as a graph, with each variable being a probability distribution rather than a single value. At the top level (Figure 11.1) the model includes the demerit point concept. The wash performance is then 100 less the total demerit points. In turn the soil demerit points is the sum of those demerit points for patches of soil and fine particles of soil. ‘Soil.DemeritPoints.Total’ refers to the total points per item of dishware, capped at 10 points.

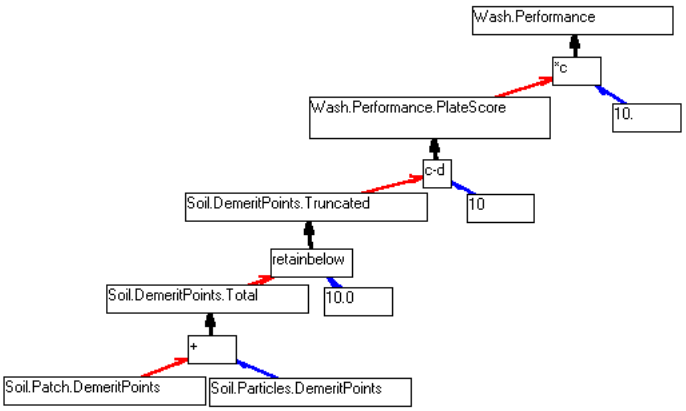


Figure 11.1: Wash performance graph.

Rather than attempt to predict particle size and then apply the ANSI/AHAM scales, the current method predicts the numbers of only two groups of particles; small (hereafter called *particles*) and large (hereafter called *patches*), and applies a different demerit weight to each. This simplifies the ANSI/AHAM scale, which is unmanageable as a simulation strategy. The mathematical representation of the figure is:

Wash.PerformancePercent =

*(10- Max[(Soil.Patch.DemeritPoints + Soil.Particles.DemeritPoints),10])*10*

Equation 11:2

This equation is substantially different to that of ANSI/AHAM. Each item can get a total of ten rather than nine demerit points. Also, the demerit points determine the wash performance directly (100-10x[demerit points]) so the weighted average method is circumvented, and with it any debate about values of weights. The term '*wash performance*' is used here to distinguish this proposed model from the ANSI/AHAM 'wash index' or 'washing index'.

Sub-graphs have been created for each of particle and patch demerit points, and these are described next.

11.4.3 **Residual Soil patches - soil removal**

This model proposes that the number of patches is an indication of *effectiveness of soil removal*. Therefore, those factors believed to affect soil removal are worked into this part of the model. Some of these factors are the soil type and condition, wash time, and the sprayer system. These and other factors are shown in Figure 11.2 along with the processes used to incorporate them.

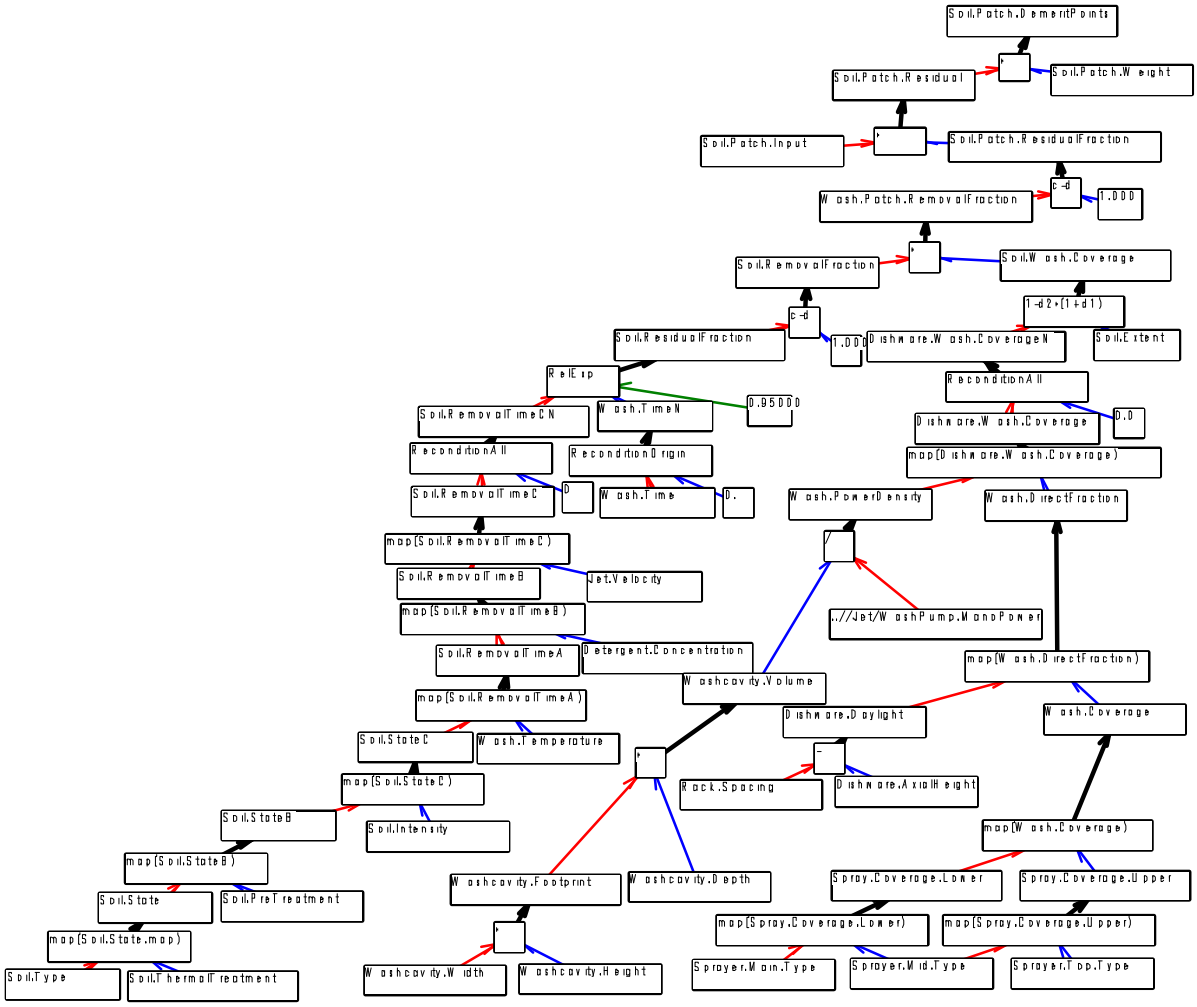


Figure 11.2: Graph for number of demerit points for patches of soil.

Some parameters such as ‘Washcavity.Volume’ (lower middle of figure) are purely quantitative and therefore use mathematical operators like product (represented by

‘*’ in the figure). Even so the inputs like ‘Washcavity.Width’ are probability distributions that may be widened or narrowed to express the degree of uncertainty in the parameter. DSI uses a probabilistic computation algorithm to process the mathematical relationships. Other relationships are qualitative, and these are shown by a ‘map(..)’ operator. A map is effectively a *decision table*, the process of which is described in previous chapters.

At the top levels the relationships are quantitative and may be expressed as:

Soil.Patch.DemeritPoints =

*[Soil.Patch.Input - (1 - (Soil.Removal.Fraction * Soil.Wash.Coverage))] * Soil.Patch.Weight*

Equation 11.3

Below this the strategy was to predict the soil removal fraction (left branch of the figure) and the wash coverage (right branch).

11.4.4 **Residual Soil particles - rinse effectiveness**

This part of the model simulates the demerit points caused by small loose particles that have been redeposited on the dishware. Those factors believed to be responsible for redeposition and the effectiveness of the rinse process are included here. These include the dilution of soil by the successive rinses, the characteristics of the filter (if any is provided) and the initial loading of particles. These and related factors are shown in Figure 11.3 as a computational graph.

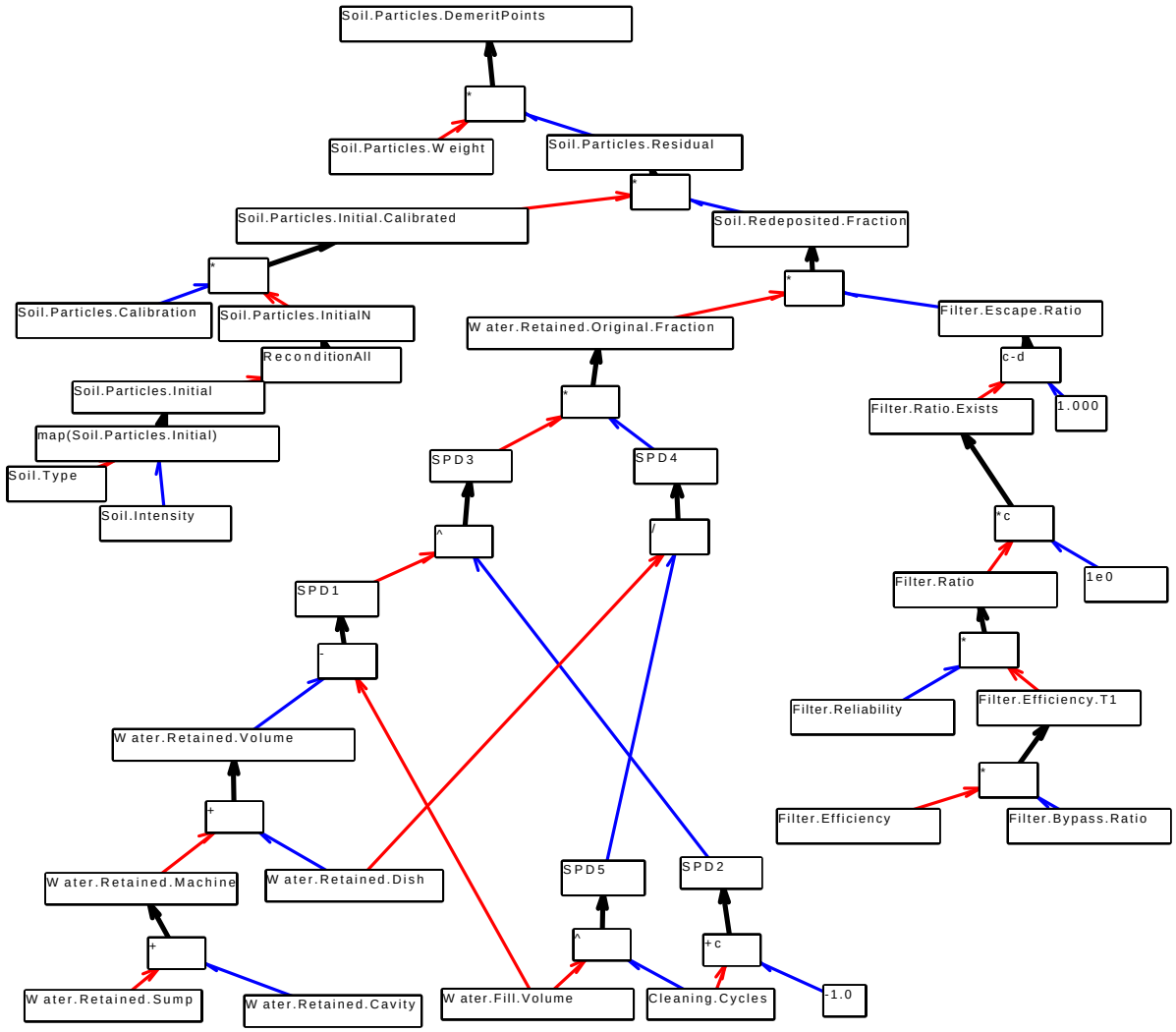


Figure 11.3: Graph for demerit points for particles.

The main branches on the graph are the initial number of particles (depends on soil type, soil intensity, and a calibration constant), the fraction of water retained (depends on volumes and number of wash cycles), and the fraction of water that is unfiltered (depends on filter characteristics).

11.4.5 Probabilistic computation of maps

DSI provides probabilistic computation on parameters that are probability distributions rather than single point deterministic values (with the exception of some constants). The DSI methodology provides combinatorial processes for the mathematical operators (plus, minus etc.) and a map (or decision table) process for the qualitative relationships.

Maps make up a significant part of the wash model, which is to be expected given the lack of quantitative understanding about the fundamental wash mechanisms. The interested reader is referred to the following chapter for further details.

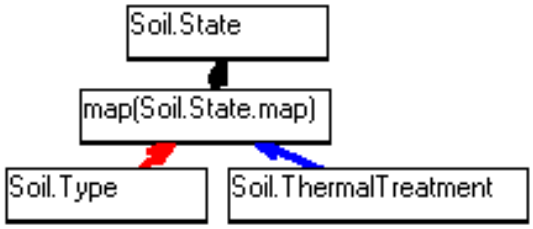


Figure 11.4: Representation of a map relationship in the DSI software.

The map takes two qualitative parameters, such as Soil type and Soil thermal treatment and combines them to produce the output. The DSI software represents the relationship as a graph, as per Figure 11.4.

For Soil type, the ANSI/AHAM test specifies the type of soil, and the quantity. This soil profile is shown in Figure 11.5, where the proportions have been determined from the relative masses of the soils. The advantage of

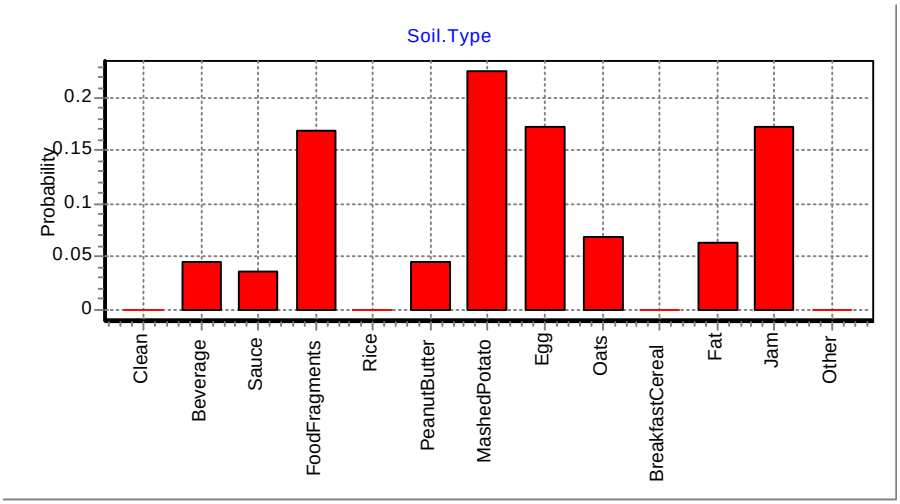


Figure 11.5: Soil type shown as a histogram based on mass of various soils used in the ANSI/AHAM test.

the DSI method is that other soil profiles may easily be set up. For example, the ANSI/AHAM profile ignores rice. A new profile can easily be asserted and propagated through to determine the resulting wash performance.

The model accommodates various thermal treatment, as shown in Figure 11.6. In the ANSI/AHAM test the soil is air dried onto the dishware, in which case the model uses soil thermal treatment of Dried (with a probability of one, i.e. certain to be nothing else). This parameter permits other profiles to be defined to suit user habits.

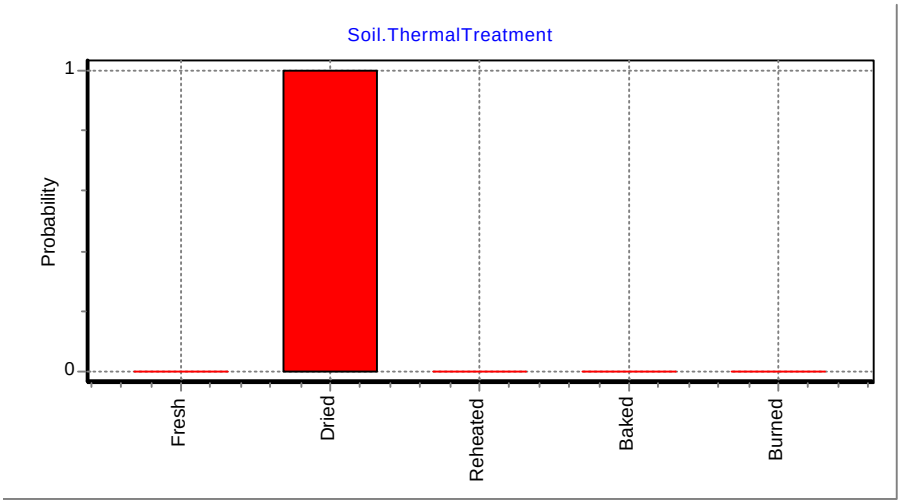


Figure 11.6: The profile for Soil thermal treatment can range from 'fresh' to 'burned' soil. For the ANSI/AHAM test there is only a single value: 'dried'.

Soil State is computed by applying a subjective map to soil type and soil thermal treatment. A fragment of this map is shown in Figure 11.7. The map process corresponds to a decision table (see an earlier chapter for details).

Soil.State			Soil.Type												
			Clean	Bever	Sauce	FoodF	Rice	Peanut	Mashe	Egg	Oats	Break	Fat	Jam	Other
Soil.Therm	Fresh	Comment													
		Clean	1	0	0	0	0	0	0	0	0	0	0	0	0
		SolubleFilm	0	0.333	0.1	0	0	0	0	0.5	0	0	0	0.5	0.142
		FineParticulate	0	0.333	0.2	0.25	0	0	1	0	0.333	0.333	0	0	0.142
		LoosePieces	0	0	0.2	0.5	1	0	0	0	0.333	0.333	0	0	0.142
		StickyPaste	0	0	0.3	0.25	0	1	0	0.5	0.333	0.333	0	0.5	0.142
		Greasy	0	0.333	0.2	0	0	0	0	0	0	0	1	0	0.142
		DriedStuck	0	0	0	0	0	0	0	0	0	0	0	0	0.142
		CookedOn	0	0	0	0	0	0	0	0	0	0	0	0	0.142
		BurnedOn	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11.7: Map to convert soil type and soil thermal treatment into soil state.

The output of the map process is ‘soil state’ as shown in Figure 11.8. It is important to note that all the inputs and outputs are qualitative, so the process may be used for relationships that are qualitative.

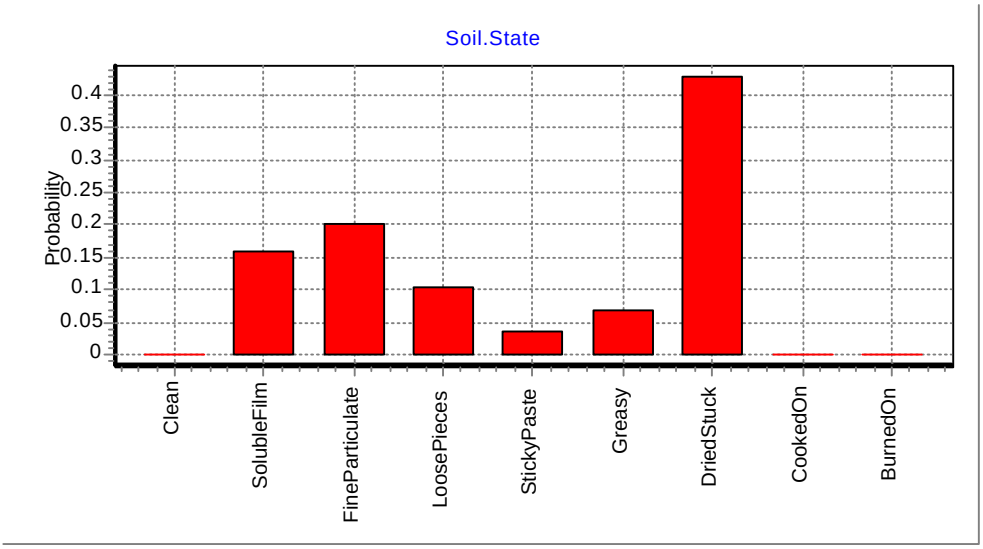


Figure 11.8: Result for Soil state, as a combination of soil type and soil thermal treatment.

11.4.6 Probabilistic computation of arithmetic operators

For the sake of completeness the DSI probabilistic computation of an arithmetic operator is illustrated here. Further details are available in earlier chapters. The example uses the operation shown in Figure 11.9, being a fragment of the bigger computation graph for soil particles demerit points. The water retained in the machine at the end of a wash cycle is required, and it is the sum of the water retained in the sump and the water retained as a film on the wash cavity. Both inputs are represented by probability distributions, namely Normal(mean 0.250, standard deviation 0.02) and Weibull(characteristic life 0.1, shape factor 2) respectively. The result after a probabilistic addition is shown in Figure 11.10. The horizontal axis is volume (litres) and the vertical scale represents probability as histogram. The individual data points are not shown, but the distributions have different bin widths so they scale differently in the vertical axis.

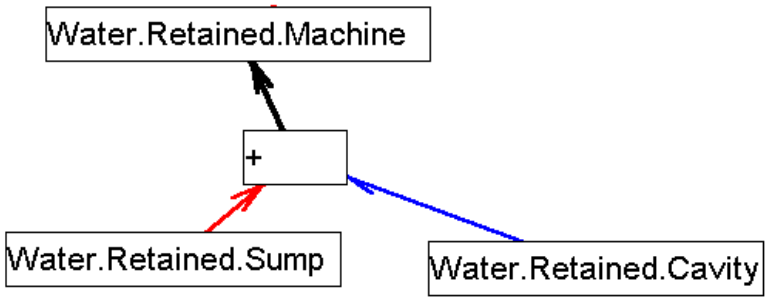


Figure 11.9: Probabilistic addition in DSI.

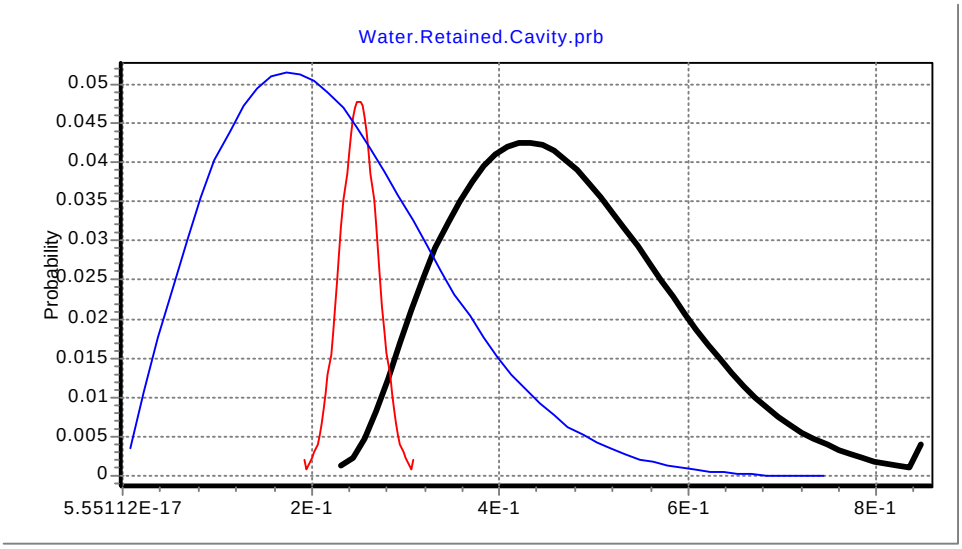


Figure 11.10: Result (heavy line) after addition of two input probability distributions (light lines).

11.5 **Calibration of wash model**

A dishwasher with known wash performance was used to calibrate the model. This was the ASKO 1805. Figure 11.11 shows a view of the inside. This product has a conventional layout, with rotating spray arms at base and middle, auxiliary fixed jets at top, and no filtration other than a filter plate. At the bottom is the main rack for dishes, plus a cutlery basket (bottom centre), which takes up about two thirds of the cavity. The top third is taken up by a second rack which accommodates cups, glasses and smaller dishware such as saucers. Against the roof of the cavity, at top left, is a small third rack suitable for items such as long knives.

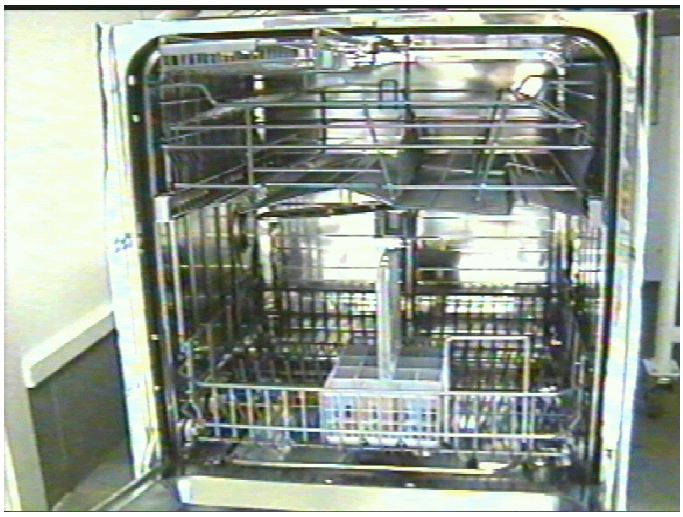


Figure 11.11: Interior view of ASKO 1805 dishwasher.

Two ANSI/AHAM wash tests for this product were available courtesy of Gin (2000). The glass and cutlery wash data were ignored as other wash effects are suspected to be operating with those ware.

To convert the wash results to the current model, it was necessary to first stretch the raw scores onto a 0-10 scale (instead of 0-9). The wash performance is then determined directly, either by averaging all the scores, or more usefully by plotting the data as a histogram (Figure 11.12). The diagram shows how variable the wash performance is. The mean wash performance is 38%. (The same data give an ANSI/AHAM wash index of 42%).

It is apparent from the wide spread of the observed data that wash performance is a highly uncertain parameter. Note also that the standard deviation of demerit points is a large fraction of the mean value, confirming the variability of the process. It may be that the ANSI/AHAM test has an over-sensitive scoring method, and/or dishwasher performance is intrinsically variable. The ANSI/AHAM method does not provide a measure of this variability. Furthermore, by averaging the frequencies of each score rather than averaging raw scores themselves, the ANSI/AHAM method suppresses some of this variability.

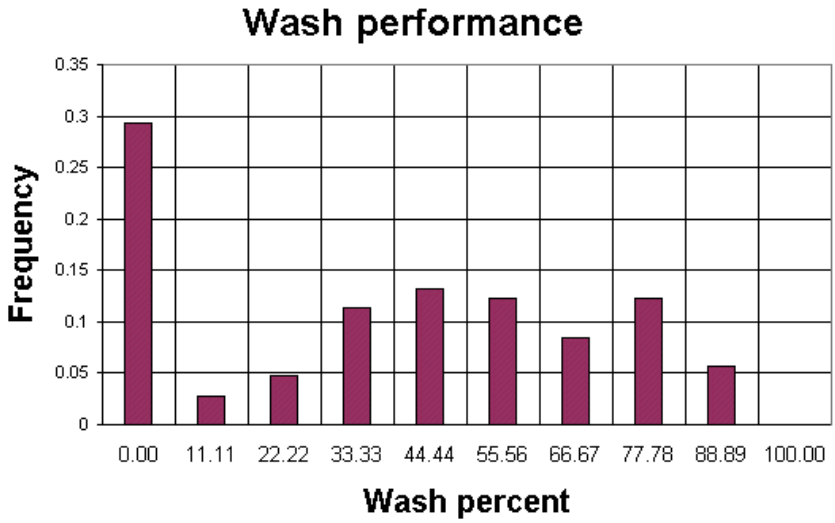


Figure 11.12: Actual wash performance for the ASKO 1805.

The wash performance model was then adjusted to fit this wash performance, using the known characteristics of this product. In particular the parameters 'Soil.Particles.Calibration' and 'Soil.Patch.Input' are provided for calibration purposes. The resulting wash performance for this machine after running the DSI simulation is shown in Figure 11.13. The chart should be interpreted as a density or histogram that shows where the wash performance is likely to fall. Horizontal axis is wash performance as percentage. Mean is 38.1%, and the standard deviation 27.5. Although the low performance spike at left appears to be more prominent than in the observed data, the reader should note that there are more data points in the rest of the current curve due to the finer histogram structure. The probability of the spike on its own is similar in both the observed and simulated results.

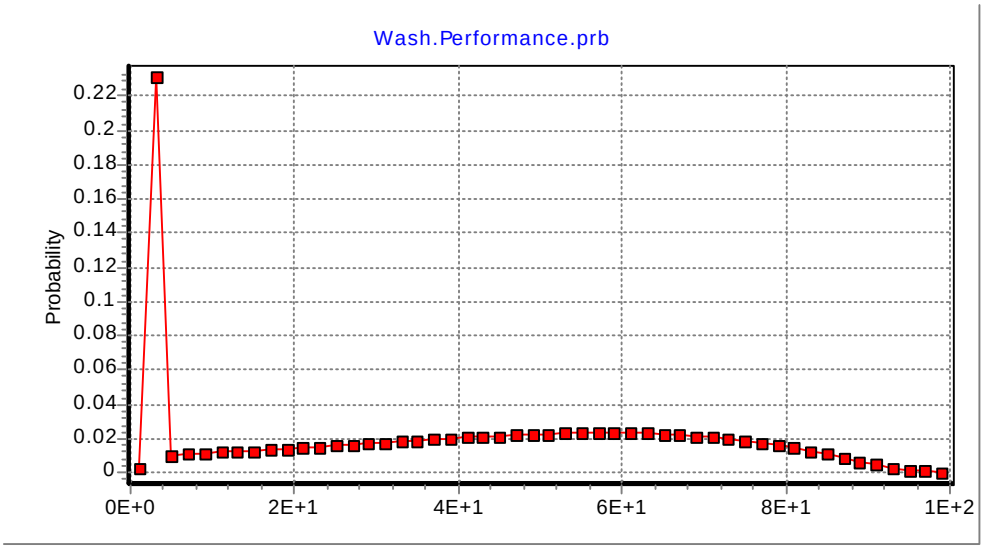


Figure 11.13: DSI predicted wash performance for the ASKO 1805.

A visual check of Figure 11.14 shows that the shape of the distribution compares favourably to the observed wash performance data and the mean is also close. There is no significant difference between the simulation and observed data at the 80% significance level using a Chi square test (see Appendix 2.4), which confirms the reasonableness of the calibration.

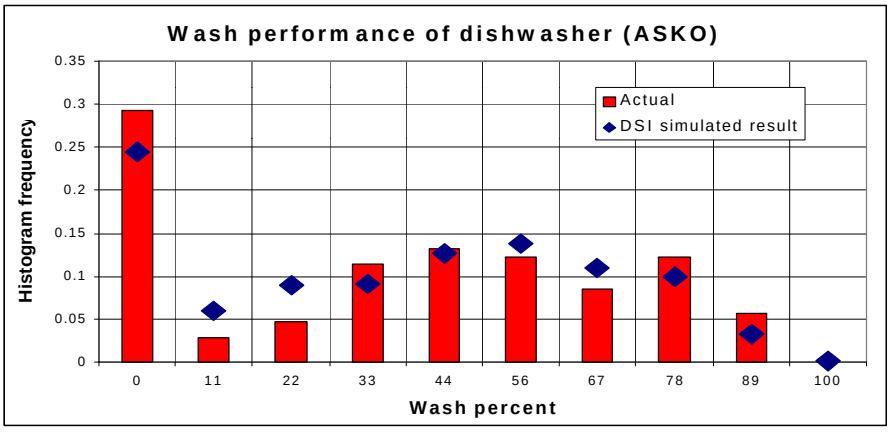


Figure 11.14: Wash performance for the ASKO 1805 with the bars showing measured data, and the points showing the DSI simulation results.

This demonstrates that it is possible to successfully calibrate the method to model the considerable uncertainty in wash performance, the shape of that uncertainty, and important statistics such as the mean.

11.6 **Predicted wash performance of a different machine**

The DSI model was next used to predict the wash performance of a different dishwasher. This tests the robustness of calibration. The selected dishwasher was a GE Profile GSD4330 for which data were supplied by Gin (2000). The same DSI model as before was used, with the only change being different assertions on spray arms: the sprayer configuration was changed to a coarse spray at the middle of the cavity, and a spray arm at top. The results of the simulation are shown in Figure 11.15.

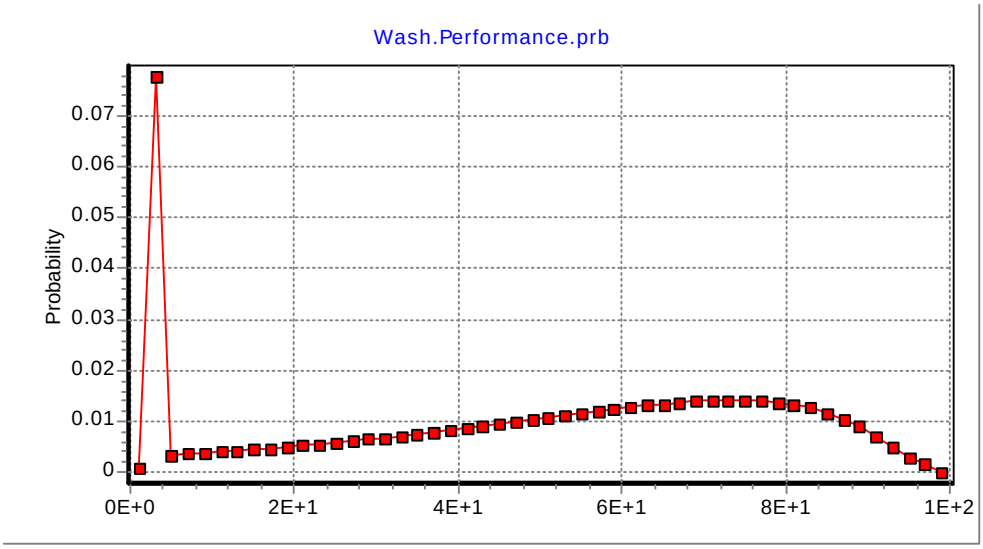


Figure 11.15: Simulated wash performance for the GE machine.

The mean wash performance of 48.6% compares favourably with that of 47.5% (standard deviation 31.9) for the observed data (the average of the raw scores after stretching onto a 0-10 scale).

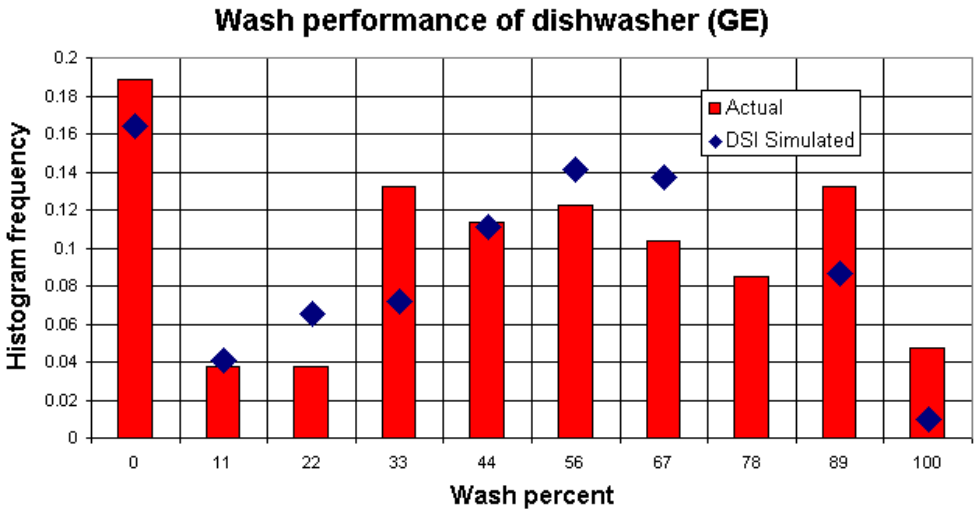


Figure 11.16: Wash performance for the GE Profile with the bars showing measured data, and the points showing the DSI simulation results.

The actual and simulated results are overlaid in Figure 11.16 to ease comparison. It is evident that the fit is not as good as for the ASKO calibration case. However the simulation predicts the mean with reasonable accuracy, and gives some indication of the shape, despite the highly qualitative nature of the model.

It might be concluded that this sprayer configuration is superior to that of the ASKO product, for wash performance. However this is not necessarily the case, as there could be other effects that have not been captured by the model. All that can really be said, given the highly qualitative relationships in the model, is that this sprayer configuration might be worth investigating as a design option. This demonstrates the value of the DSI methodology as a steering mechanism towards design refinements that could be rewarding. It also demonstrates that the system has resolution powers to be able to distinguish small (and qualitative) changes in geometry configuration.

11.7 **Predicted Wash performance under other soil conditions**

Any standard test such as ANSI/AHAM aims for consistency rather than necessarily attempting to reproduce all usage conditions. Also, some foods are introduced simply because they provide a challenge to remove, not because people eat them to that extent. For example some tests (not ANSI/AHAM) use ground spinach, probably because it makes a clear marker for redeposited soil and not because families with children eat a lot of it! The wash performance model developed here may be used to explore other usage profiles. It may be particularly valuable to change the profile for say soil type, to explore how the machine would operate in a target market where different foods were expected. The food profile used in this simulation (Figure 11.17) contains changes such as nominally clean dishes, inclusion of rice, less peanut butter, and breakfast cereal rather than oats, among others.

Running the simulation on the ASKO model with this change in soil profile gives the result shown in Figure 11.18. The mean wash performance is now 41.6, which is an increase over the result for conventional soil profile.

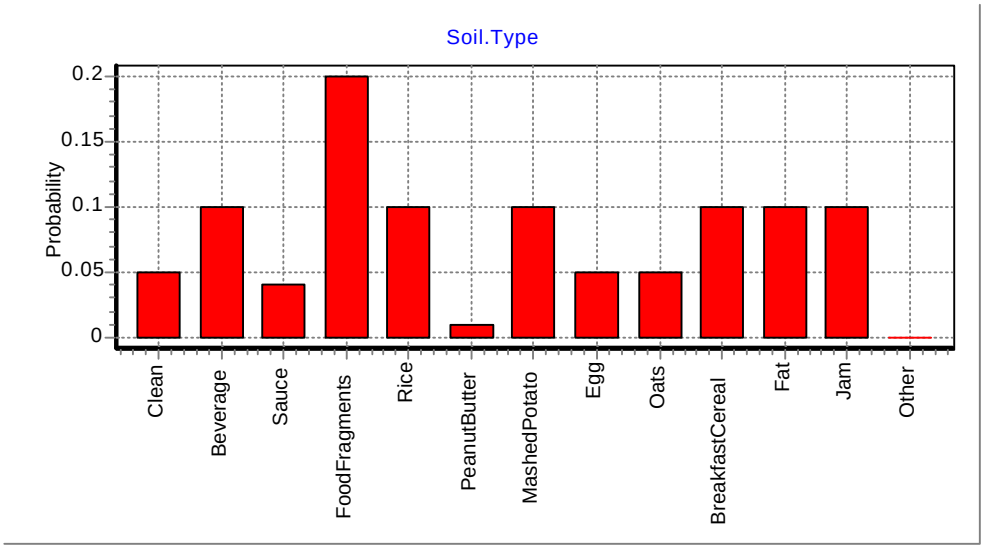


Figure 11.17: Modified food profile to simulate the profile that a typical family might generate.

Although physical test results were not available to verify this simulation result, it suggests that to the extent to which real people load their dishwasher with these other foods, they could experience better wash performance than the standard tests indicate. It also suggests that a dishwasher may have different wash performance scores depending on the test method used (the tests vary in soil among other parameters).

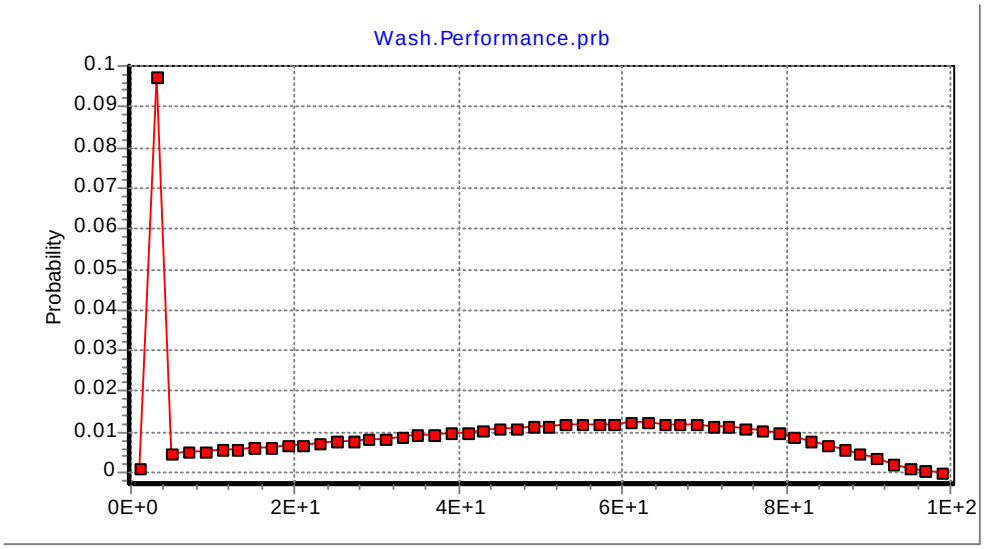


Figure 11.18: Simulated wash performance for the ASKO 1805 operating on the modified soil load.

It is important to note that a substantial part of the model is qualitative and based on opinion, and thus the accuracy of an exploration result such as this is only as good as the input data. The benefits of the DSI method compared to deterministic methods are that the uncertainty is shown and qualitative factors may be included.

11.8 **Discussion**

A significant issue with the development of this model of wash performance was that of multi-point calibration. Not only does the model need to be calibrated to fit a single point mean, but it also needs to show reasonable fit to the shape of the distribution. Matching the distribution shape adds additional degrees of freedom into the

calibration task. There are multiple parameters that can be adjusted in the calibration process, so it was not simply a matter of adjusting one or two parameters. Instead the model has to be tuned in an iterative process. Naturally the process becomes yet more complex when another dishwasher is added and the same model has to be tuned for that too, without invalidating the earlier calibration. The results presented above show that this process is possible, but it is worth noting that it is a significant part of the model development.

There is a difference between validation and calibration of a model. Validation occurs when the underlying system processes have modelled with sufficient accuracy that valid results are obtained. Calibration refers to the adjustment of a model, using factors without strong justification, so that the output of the model more closely resembles the output of the real system. The issue, as Law and Kelton (1991) point out, is whether a calibrated model is generally valid, or only accurate for particular input data. They observe that a calibrated model may be validated by using an independent real model, and comparing the model output against this new real data.

In the work that has been presented here a probabilistic simulation was developed for the highly qualitative parameter of wash performance. The model was successfully calibrated. It also had defendable accuracy in predicting the performance of a different machine, and this provides some validation of the model too. It should be noted that certain regions of the model (for example the filtration) have not been exercised. It would be another project in itself to explore and calibrate every part of the model. This is beyond the scope of the current project, since the aim was to demonstrate that a model of wash performance could be created and defended, and that large uncertainties could be propagated through it. This much has been achieved.

Validating a qualitative model, such as that presented here, is likely to be difficult if not impossible, since the model depends on expert opinion. Any number of experts (with different opinions) may potentially be found. However even a subjective model

may be better than no model, and the examples above show how the model permits the design space to be explored, even in the qualitative domain.

The DSI method requires that an expert's knowledge be expressed systematically, i.e there must be some rational basis for the beliefs. In turn this permits the expressed knowledge to be scrutinised and even debated between experts, which would appear a positive feature.

It is important to note that the DSI method accommodates uncertainty in two dimensions. It does not require the expert to be certain of belief, but provides a map with probability weights for various outcomes. This has been referred to as uncertainty of analysis. For mathematical relationships there is no uncertainty of analysis. DSI also accommodates process variability, which is the uncertainty in the asserted variables. The outcome of a DSI computation is affected by both forms of uncertainty.

11.9 **Conclusions**

This project used probability distributions and expert belief, with the DSI probabilistic engine, to create a model of dishwasher wash performance. The model incorporates the high uncertainties typical of early design. Uncertainties of relationship (ie qualitative relationships) as well of measurement (eg quantitative process variability) were included, using the DSI method. The qualitative relationships were modelled with maps, the contents of which were based on beliefs (i.e. opinions) with their uncertainty. Two principal mechanisms were proposed to contribute to wash performance: *effectiveness of soil removal*, and the *effectiveness of the rinse process*. In other words, one part of the problem of wash performance is getting the soil off the dishware, and the other part is preventing it from being redeposited after it has been loosened. The model was calibrated against wash data from a known machine, and then predicting the wash performance of a different brand of machine. Probabilistic computation was used throughout. Subsequent projects may be able to

extend the validity of the dishwasher model by incorporating new qualitative and quantitative knowledge.

The results demonstrate that the DSI methodology can assess qualitative and quantitative performance at the early design stages where relationships are uncertain and information is sparse. Being able to assess uncertainty is potentially an important management tool for the early design stages. For example a parameter with large uncertainty shows the level of risk in the design, and the extent to which issues have been solved in the design process.

Chapter 12

Development of a Wash model

This chapter explains the development of a model that is able to simulate the wash performance of a dishwasher in probabilistic terms, using qualitative and quantitative information.

12.1 **Wash performance**

Wash performance is measured as the amount of soil left on dishes after washing. In this project a new wash performance measure was created. This chapter describes the details of the model, in particular the probabilistic computation that lies behind it.

As the earlier chapter described, the model consists of two main branches, being removal of patches of soil, and prevention of redeposition of soil. These are combined at the end to determine the overall wash performance. At the top level the model for wash performance is described in terms of Figure 12.1 (being a repeat of the figure from the earlier chapter for convenience).[156] Those two main branches are described next.

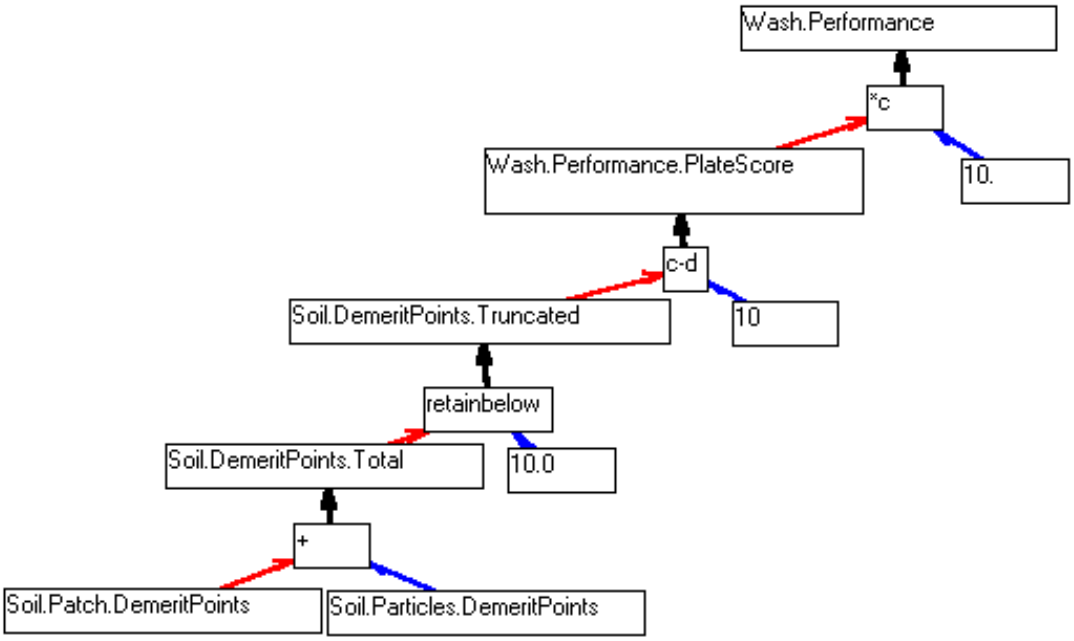


Figure 12.1: Wash performance model.

[156]The wash performance is 100% less the soil demerit points. In turn the soil demerit points is the sum of those demerit points for patches of soil and fine particles of soil. 'Soil.DemeritPoints.Total' refers to the total points per item of dishware. This is capped at 10 points . Subsequent calculations determine the wash score per dishware ('c-d' operator) and then the Wash Performance as a percentage.

12.2 **Model details for Residual Soil patches - soil removal**

The model for demerit points (per ware) due to soil patches is shown in Figure 12.2 (repeated from previous chapter). Next the various aspects of this model are discussed, starting at the lower left. In the anticipation that to reproduce all the data in print at this point would be unhelpful to the reader, especially as the maps are large tables, the discussion is directed towards the reasons for various features in the model and not so much a listing of the data.[157]

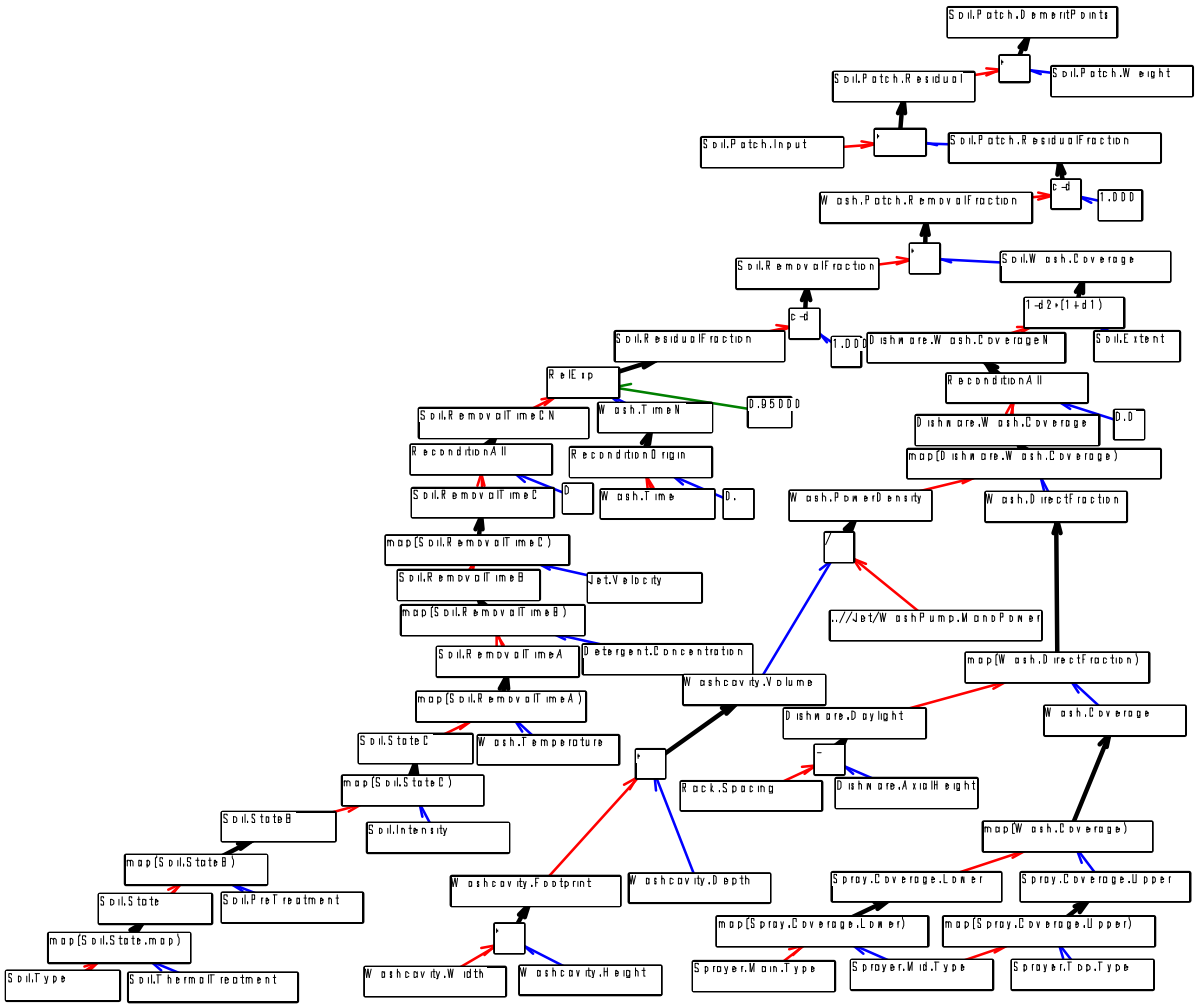


Figure 12.2: Graph representing demerit points for patches of soil.

[157] The full data are available on CD-ROM from the University of Canterbury, and can either be viewed with the DSI software (a copy of which is also provided) or with a standard text editor, word processor or spreadsheet software.

The combination of Soil type and Soil thermal treatment using <Soil.State.map> to produce Soil State was discussed in the previous chapter to which the reader is referred. The wash performance tests use soil that has dried onto the ware. This is a realistic usage arrangement, since people tend to accumulate soiled articles until a sufficiently full wash load is obtained, and there is potential for drying to occur during this time.

Soil pre-treatment

This parameter refers to whether or not a user rinses dishware before placing in the dishwasher. Castro (1999) reports on various surveys of the fraction of people who pre-treat their dishes with a rinse, and those who load untreated dishes into their washers, such as a 1995 Soap & Detergent Association finding of 79% pre-treat and 21% no treatment, and another un-named 1984-1989 survey of 66% treat and 34% no treatment. Based on these surveys Castro proposes that weights of 60% pre-treat and 40% no treatment should be used in the energy tests. The reason rinsing is important in energy tests is that some dishwashers are equipped with turbidity sensors, so they perform fewer wash cycles or reduce the water fill if they detect less soil. Reduced cycles or water usage means that the thermal input for heating the water is also reduced, and therefore energy usage is lower.

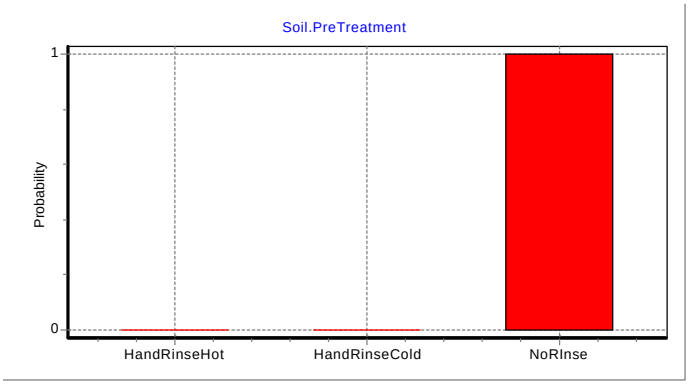


Figure 12.3: Soil pre-treatment with no rinse.

The wash performance tests such as ANSI/AHAM test provide for a standardised soil load and therefore rinse is inappropriate. Therefore ‘No rinse’ is used in the current model, though provision is made for cold and hot rinses as well.

Soil pre treatment and the previous Soil state combine using <Soil.StateB.map> to determine Soil state B. The map provides for rinses to remove some of the particulate and soluble soil, but minimal effect on the greasy and cooked on soil.

Soil intensity and Soil state C

The model makes provision for three soil intensities: light, medium and heavy. The ANSI/AHAM loading is assumed to correspond to medium intensity. The intensity is combined with Soil state B in a map to produce Soil state C. The effect of the map is to move some of the soil to a tougher state if the soil is heavy. There is no change to soil state when the soil intensity is medium (as is the case in the model). However the model provides for the more general case where the soil-thermal treatment, -pre treatment and -intensity differ from the ANSI/AHAM test.

Soil removal time A, and Wash temperature

Up to here the model has been developed along the lines of soil state. At this point it changes focus and starts to simulate time required to remove soil. The first addition is Wash Temperature, which for many machines is fixed somewhere in the range 45-75 deg C, at least for the normal wash cycle. The ANSI/AHAM test does not prescribe a wash temperature, but simply uses the normal cycle provided by the manufacturer. The current model uses the profile shown in Figure 12.4. The distribution is wider than that simply due to process variability, and reflects the uncertainty (lack of knowledge) about what the final temperature will be.

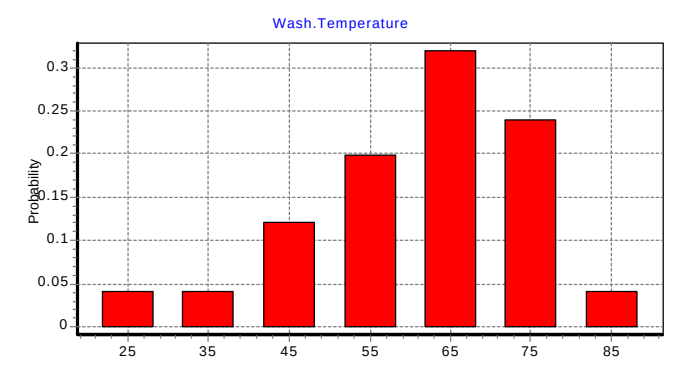


Figure 12.4: Wash temperature profile.

Hotter water temperatures generally give better soil removal, as the hot water penetrates the soil and loosens it. However water temperature directly affects the energy consumption of the machine, at least for those majority of machines that

provide their own heating. The energy consumption is a conspicuous label on the product, and it affects purchase decision. Therefore there is pressure on manufacturers not to use high water temperatures in their designs. Most dishwashers also provide for lower temperature wash programmes. These economise on energy usage and are often termed “economy” cycles. Some manufacturers provide machines with hot-fill, which means that the dishwasher does not heat water itself, but relies on receiving an external hot water supply. This permits the manufacturer the economy of omitting the water heating devices, though there is need for both hot and cold inlet hoses. Some users have ample hot water supplies, eg from central heating, and hot-fill permits them to use those economies of scale. In most cases domestic hot water heaters are set at around 45-75^{o}C, so the use of hot-fill does not necessarily give higher temperature washes. Some manufacturers provide a heating element in their hot-fill machines, so that it can be used on pure cold fill if necessary.

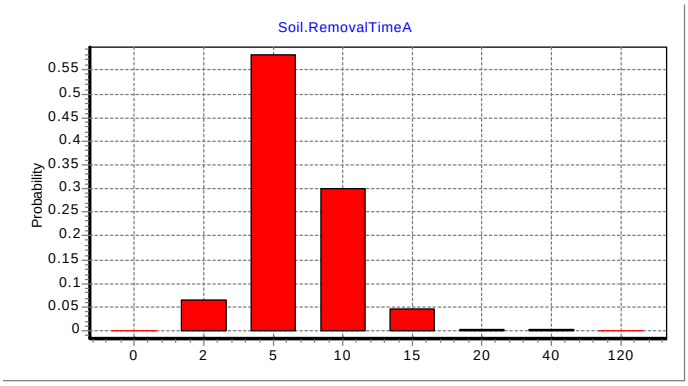


Figure 12.5: Soil removal time A.

The map <Soil.RemovalTimeA.map> converts the soil state into a time required to remove that soil, based on the wash temperature. Soil states that tend towards being baked on are given higher removal times. Hotter wash temperatures decrease removal time, but not necessarily in a linear fashion as baked on soils are modelled as little affected by even the hottest water. The result for Soil removal time A is shown in Figure 12.5. This is minutes of washing required to remove the bulk of the soil

Detergent concentration and Soil removal time B

Detergent concentration (Figure 12.6) as grams detergent per litre of wash water, is combined in a map to produce Soil removal time B (not shown). The map reduces the soil removal time as the detergent concentration increases.

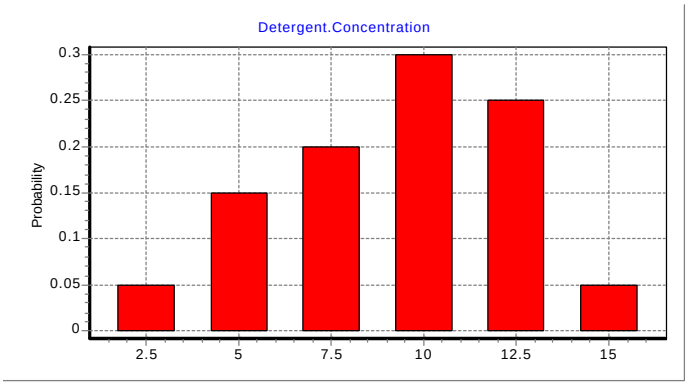


Figure 12.6: Detergent concentration.

Jet velocity and Soil removal time C

The Jet velocity (Figure 12.7) is a measure of scrubbing action of the water, and is combined in a map to produce Soil removal time C (not shown). In general dishwashers are not particularly good at removing baked-on soil as found on cooking ware that has been through an oven. This is because a more vigorous mechanical scrubbing action is required, which is not provided by the water jets.

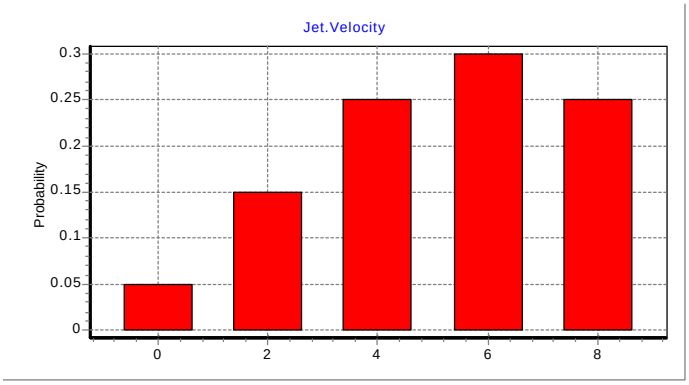


Figure 12.7: Jet velocity, in m/s.

It is relevant to note that Soil removal time C is treated qualitatively in the model: though it uses numbers, they are treated as text. To convert them into a numerical interpretation, the model uses the ‘ReconditionAll’ function,

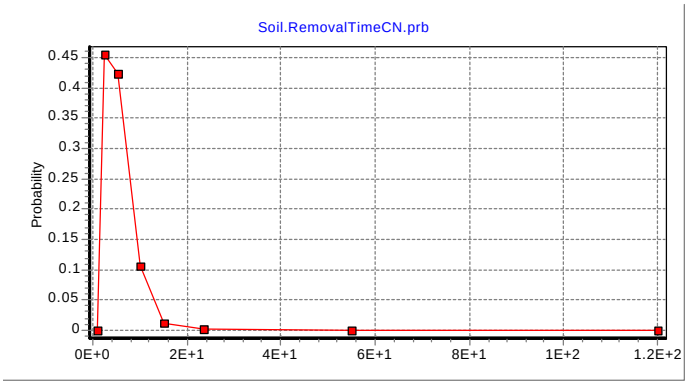


Figure 12.8: Soil removal time after converting to a numerical scale.

which assigns a numerical origin and creates intervals so that the scale is linear[158]. The results are shown in Figure 12.8. There is a small possibility of a long wash time required to remove persistent soil.

Wash time

This parameter has units of minutes in this model. Sufficient wash time is required to erode away the adherent soil. However increases in wash time are not expected to continue to improve wash performance indefinitely. Wash time is shown in Figure 12.9.

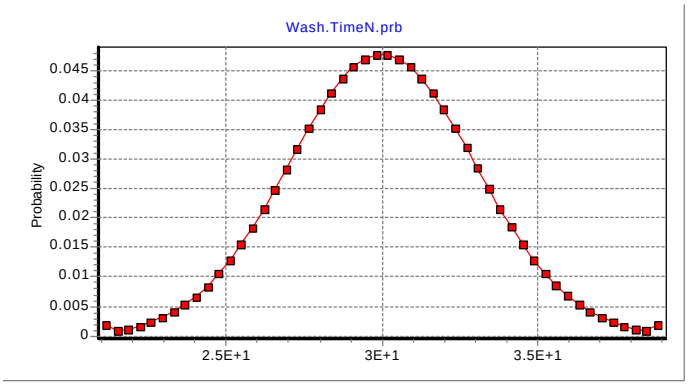


Figure 12.9: Wash time.

Soil residual fraction

Up to here the model has determined a time required to remove the bulk of the soil, and a wash time. Each is expressed as a probability distribution. The next stage combines them to determine the fraction of soil that remains at the end of the wash cycle. However a simple dominance is not used but rather a type of exponential decay function. Consequently, even if the wash time was to equal the nominal soil removal time there will still be some soil (fraction p) remaining. The equation used is

$$R(t) = \exp\left(\ln(p).\frac{D2}{D1} \right)$$

where

R(t) Soil residual fraction

[158]The 'ReconditionAll' function is not compulsory, as a textual file may be cast directly into a numerical relationship. However in doing so the origin on the numerical scale may be ill-defined and give computation warnings. The 'ReconditionAll' function ensures that the origin is consistent with the rest of the data, creating an origin if necessary.

D1 input distribution one: resistance at the p percentile, which is Soil removal time CN

D2 input distribution two: exposure time, which is Wash time N

p fraction (percentile), which is the fraction of soil that will remain when resistance D1 equals the stress D2. In this model p = 0.85 was set during calibration[159].

Each of D1 and D2 take on multiple values in a combinatorial manner, so a distribution results for soil residual fraction, not a single point. The result is shown in Figure 12.10. The steps are unavoidable artefacts produced by the relatively coarse (few data points) in Soil Removal Time CN, which had a textual origin. The steps are rounded by the variable Wash time N, which if it were crisp would have resulted in crisp steps.

Soil removal fraction

The soil removal fraction is one less the soil residual fraction. It indicates the percentage of soil patches that will be removed from the dishware during the wash process. Removal in this sense does not mean that the soil is removed from the wash cavity, instead it is removed from the dishware and may be available to be redeposited later.

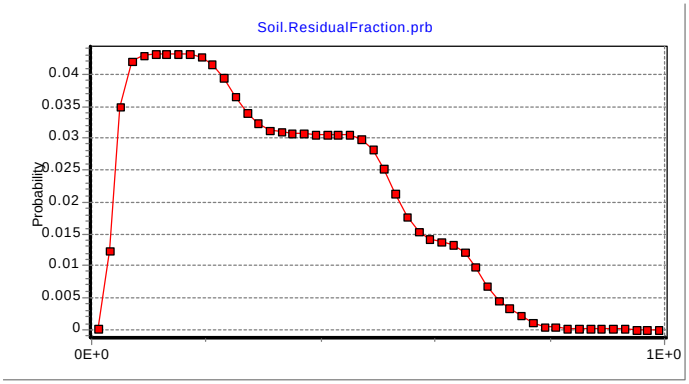


Figure 12.10: Soil residual fraction.

[159]The equation is derived from the exponential reliability $R(t) = \exp(-\lambda.t)$. If the p^{th} percentile time t_p is known, then $p = \exp(-\lambda.t_p)$ gives $\lambda = -\ln(p)/t_p$. Back substituting gives $R(t) = \exp(\ln(p).t/t_p)$. Replace t_p with D1, and t with D2 to return the equation as stated.

Wash coverage

It is now necessary to return to the parameters that will give rise to Soil wash coverage. The first of these are the type of sprayers used. Provision is made for different main, mid and top spray systems. Figure 12.11 shows the main sprayer selected in this model. However uncertainty in a design could be accommodated by giving some of the other named systems some of the probability.

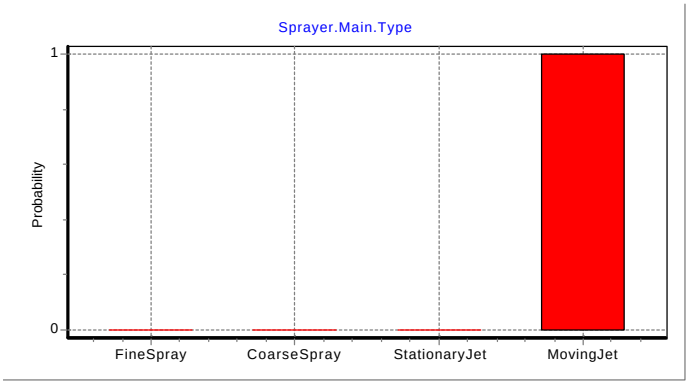


Figure 12.11: Sprayer main type is a moving jet for most dishwashers.

The wash coverage of the three spray systems is determined by a series of maps, and the result is shown in Figure 12.12.

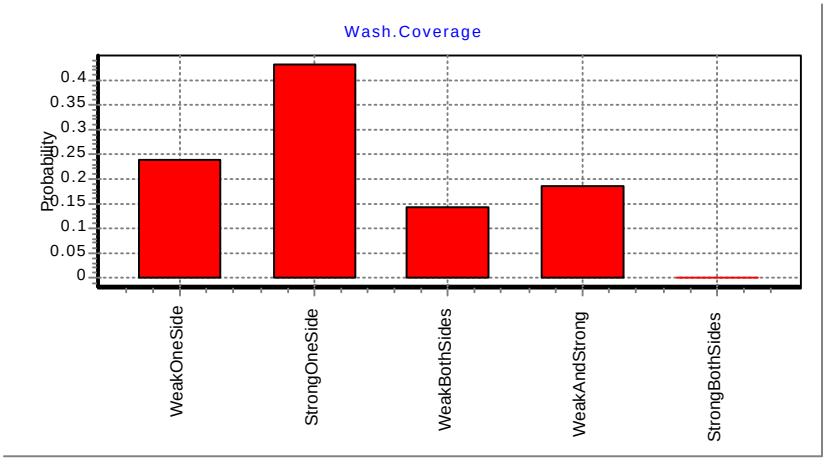


Figure 12.12: Wash coverage is determined from the three sprayer systems.

Dishware daylight

This parameter is the amount of clearance between the ware. For flat plates of negligible thickness the dishware daylight would be the same as the spacing of the rack that holds them (assuming ware is approximately vertical). For curved bowls this clearance is often completely consumed, and consequently no jet of water can directly impinge on the centre of the bowl. This is not necessarily a total failure, as

water can still reach these areas by bounce and dribble methods. Dishware daylight is calculated as the difference between rack spacing and axial height of dishware. The latter is the thickness of the box bounding the ware. The result is shown in Figure 12.13 but the inputs are not illustrated here.

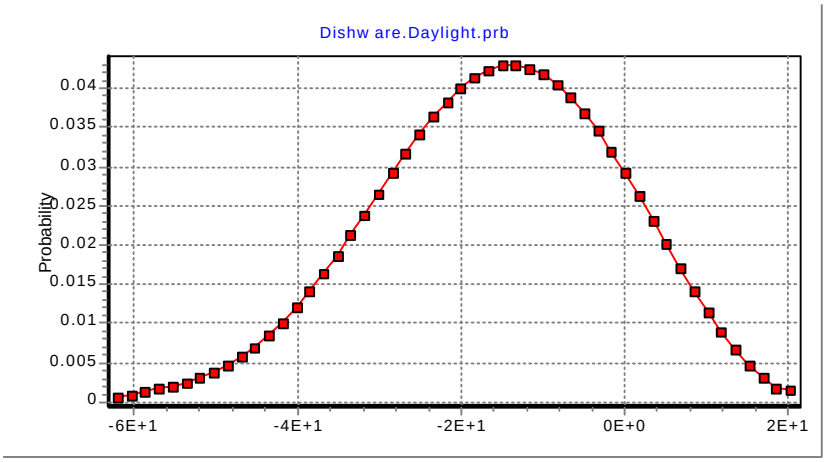


Figure 12.13: Dishware daylight.

Wash direct fraction

This is the combination of the dishware daylight and the wash coverage, using a map. The map produces a high wash direct fraction where there is positive dishware daylight and extensive wash coverage. The result is shown in Figure 12.14. Being a map operation the result has a textual interpretation, even though the text are numbers. Hence the production of a histogram rather than a smooth curve.

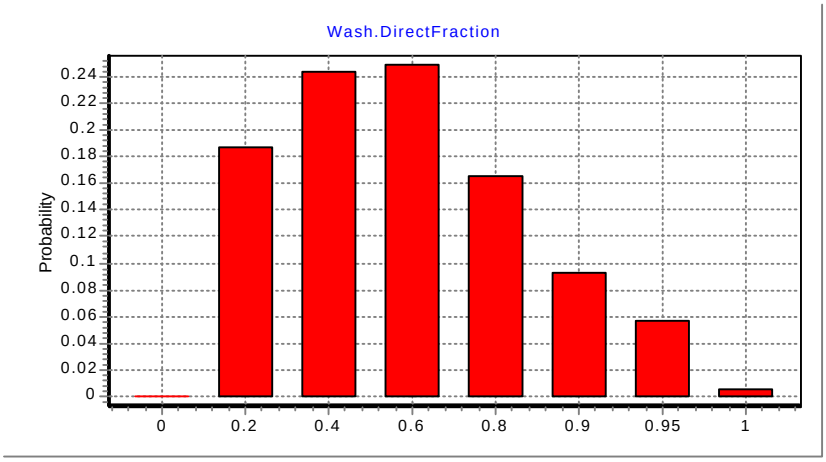


Figure 12.14: Wash direct fraction.

Wash cavity volume

This is determined as the product of the width, height and depth of the cavity. The product operation has to be done in stages, and the end result is shown in Figure 12.15.

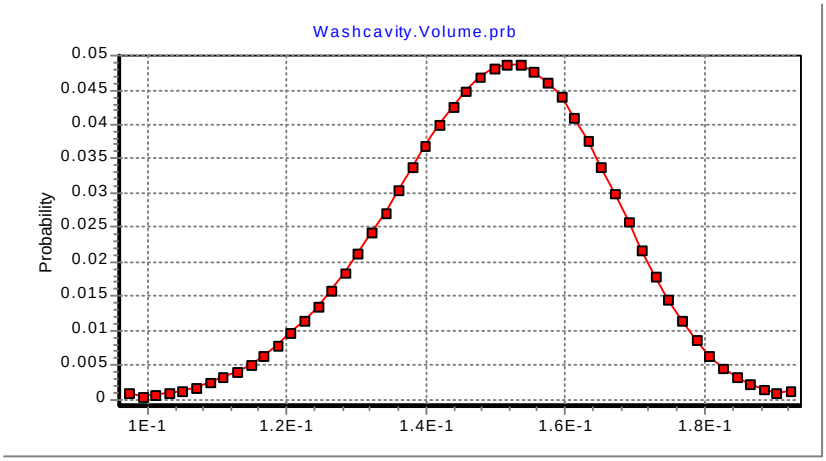


Figure 12.15: Wash cavity volume in cubic metres.

Wash power density

A separate model computes the manometric power of the wash pump given large uncertainty[160]. The wash power density is then the manometric power divided by the cavity volume, and is intended to be a parameter for the amount of splashing and indirect wetting that the ware receives. The results are shown in Figure 12.16.

[160]This model is described in further detail in a following chapter.

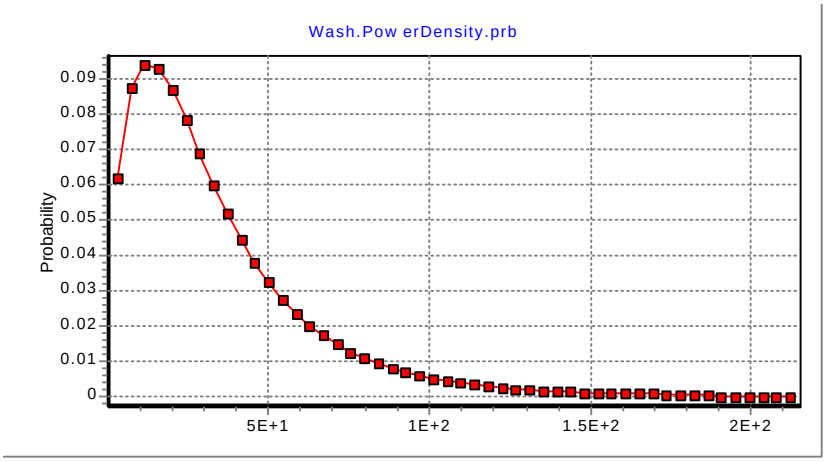


Figure 12.16: Wash power density [W.m^{-3}]

Dishware wash coverage

The dishware wash coverage is determined from a map that combines the wash power density and the wash direct fraction. The wash power density is a numerical data file, but it can be cast direct into the map as the map simply reorganises it into the histogram bins defined in the map. The output from the map will be used as a numerical scale, so as a precaution the 'ReconditionAll' function is applied to ensure that it has a consistent origin and data structure. The final result of dishware wash coverage N is shown in Figure 12.17.

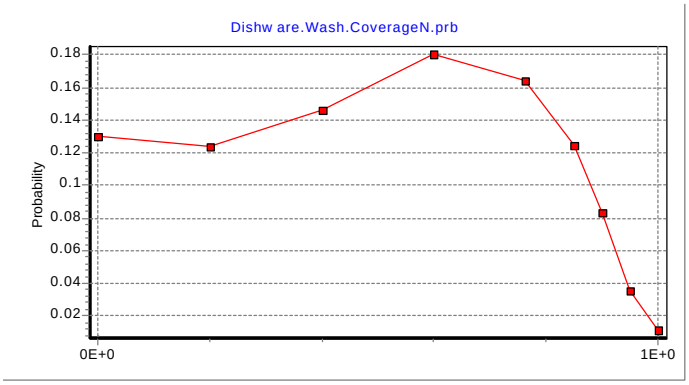


Figure 12.17: Dishware wash coverage N as a fraction.

Soil wash coverage

This is a combination of the soil extent and the dishware wash coverage N, using the operator 1 - D2*(1+D1). This is explained as a rearrangement of the ‘cleanness’ (1-soil extent) less the amount removed (Soil extent x dishware wash coverage). The result is that if the soil extent is low then there is a greater chance that it is all covered, even if the dishware wash coverage is less than 100%. The result is shown in Figure 12.18.

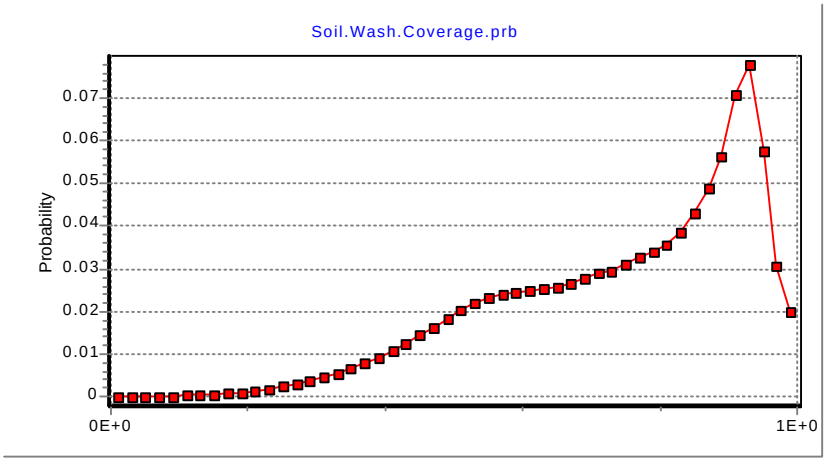


Figure 12.18: Soil wash coverage as a fraction.

Soil patch residual fraction

The Wash patch removal fraction is the product of the soil wash coverage and the soil removal fraction. Unity less that result gives the Soil patch residual fraction, which is shown in Figure 12.19.

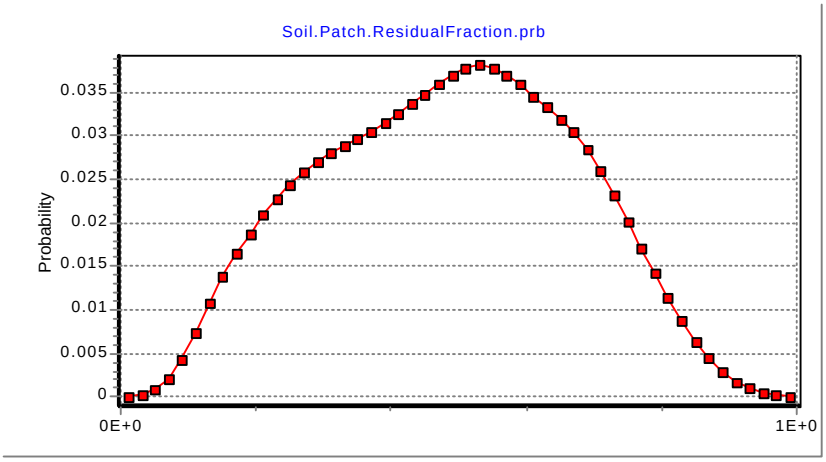


Figure 12.19: Soil patch residual fraction as a fraction.

The number of soil patches left on the ware is the product of the Soil patch residual fraction and the soil patch input (the number of soil patches in the first place). Soil patch input was set during calibration, and is shown in Figure 12:20.

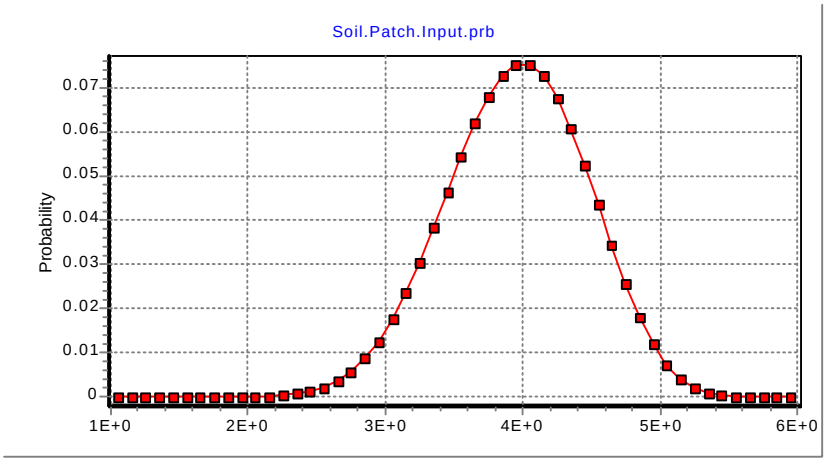


Figure 12.20: Soil patch input.

The soil patch demerit points per dishware is then the product of the soil patch residual number and the number of demerit points that each patch attracts (soil patch weight, also set during calibration and see Figure 12:21).

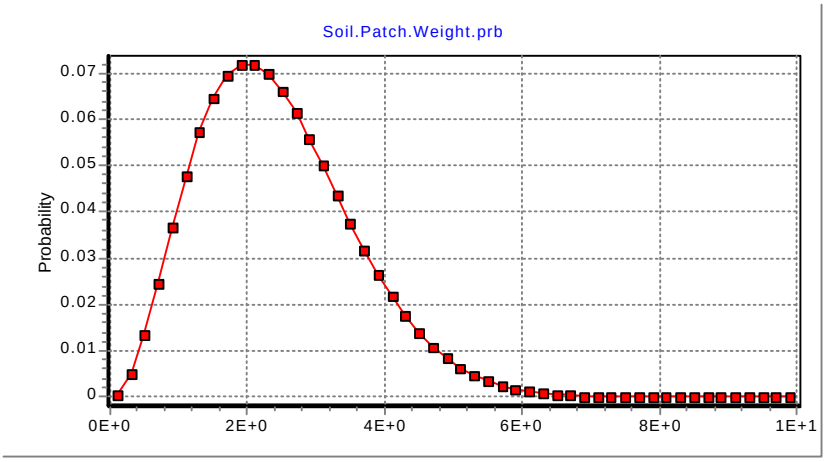


Figure 12.21: Soil patch weight.

The final result is shown in Figure 12.22, as the distribution of demerit points that a dishware item will attract.

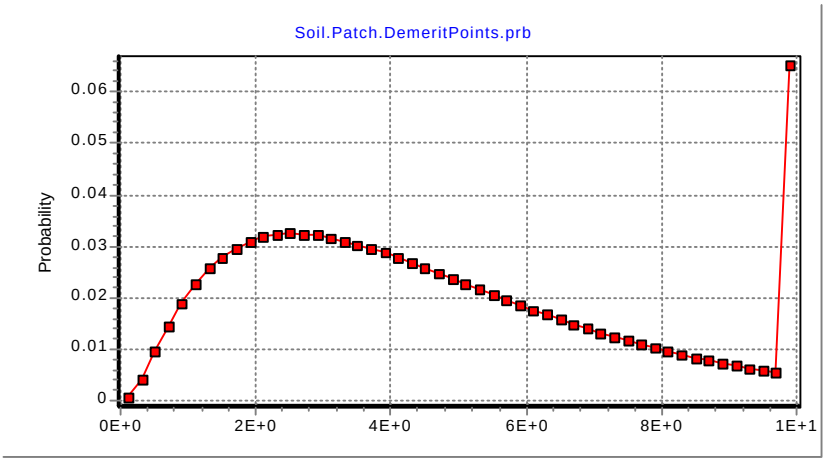


Figure 12.22: Demerit points attracted by a dishware item.

This completes the explanation of the soil patch branch of the model.

12.3 **Model details for Residual Soil particles - rinse effectiveness**

Removing the soil from the dishware is the first task in the wash process. Preventing it from being redeposited is the next. Redeposition occurs naturally, since the wash fluid is re-circulated within each wash or rinse cycle. Soil redeposition is especially a problem with non-soluble granular foods like ground spinach and mashed potato.

12.3.1 **Derivation of model for Dilution and Filtration of soil particles**

This project makes the assumption that a dilution principle is at work, and that redeposition would be reduced by decreasing the soil loading, increasing the rinse volume, and increasing the number of rinses.

Given that there are S_o fine particles to start with, and that each wash or rinse removes a portion of the particles, then the number of soil particles left in the machine after N_e wash and rinse cycles is

$$S_e = S_o\left(1 - \frac{V_d}{V_f}\right)^{N_e}$$

where

S_o initial number of particles (i.e. soil load)

V_f volume of water per fill

V_d drain volume, which recognises that not all the fill volume is able to be drained out, as fluid remains in the drain sump and as a film on dishware and the wash cavity

N_e number of wash cycles plus number of rinse cycles

The assumption in this model is that residual loose soil is suspended in the water that remains after drainage. Subsequent charges of wash or rinse water dilute this soil concentration. Also, it is implicitly assumed that all the loose soil is freed at the first wash rather than partially over several washes.

Of the total S_e particles, some are in the drain sump, and others are on the dishware. The Wash performance metric is only affected by those on the dishware. Therefore the number of soil particles on dishware is given by

$$S_{ew} = S_e \cdot \frac{V_w}{V_f - V_d}$$

where

S_{ew} number of soil particles on dishware

V_w volume of fluid as film on dishware

Filtration

The effect of filtration is to remove some of the soil particles, and prevent them from being redeposited. Most dishwashers use a coarse filter plate over the wash pump intake. The hole diameter is of the order of one millimetre. Generally this stops only the larger particles. In most case there is also an even coarser “pea-strainer”. The two screens are often in parallel as regards the fluid path, so it cannot necessarily be assumed that all the flow will pass through the filter plate. In many dishwasher designs there is provision for an automatic scrubbing of the filter plate, by placing holes in the lower spray arm so that a downwards jet prevents soil accumulating on the filter plate. Where this soil goes to other than remaining in suspension for longer is uncertain. It is therefore uncertain to what extent the filter plate actually prevents soil being ingested by the wash pump.

More sophisticated filtration systems are available in some dishwashers. They typically have a fine mesh screen and part of the fluid flow from the pump is directed through the filter. The soil so trapped is backwashed to drain at the end of the cycle, though this requires active intervention in the form of valves and other hardware.

Filtration is modelled here as dependent on two parameters, the filtration ratio (the fraction of fluid that is filtered) and the filtration efficiency (the smallest particle removed). Another parameter may also be important, the reliability of filtration, in that filters become blocked with persistent soil, so that system performance degrades over the longer term. The overall efficiency of particle removal is then given by:

$$\eta_o = F_f . \eta_f . R_f$$

where

F_f filtration ratio (the fraction of fluid that is filtered)

n_f filtration efficiency (the smallest particle removed)

R_f reliability of filtration at a given time

The full expression for the number of soil particles S_{ewf} on dishware at end of all wash and rinse cycles is therefore

$$S_{ewf} = S_o \,.\, (1 - \frac{V_d}{V_f})^{N_e} \,.\, \frac{V_w}{V_f - V_d} \,.\, (1 - F_f \,.\eta_f \,.R_f)$$

where the parameters are as previously defined. This equation may be simplified to:

$$S_{ewf} = S_o \,.\frac{V_w}{V_f^{N_e}} \,.\, \frac{(V_f - V_d)^{N_e}}{V_f - V_d} \,.\, (1 - F_f \,.\eta_f \,.R_f)$$

$$= S_o \,.\frac{V_w}{V_f^{N_e}} \,.\, (V_f - V_d)^{N_e - 1} \,.\, (1 - F_f \,.\eta_f \,.R_f)$$

The equation is then represented in the probabilistic computation model shown in Figure 12.23 (repeated for convenience from the previous chapter).

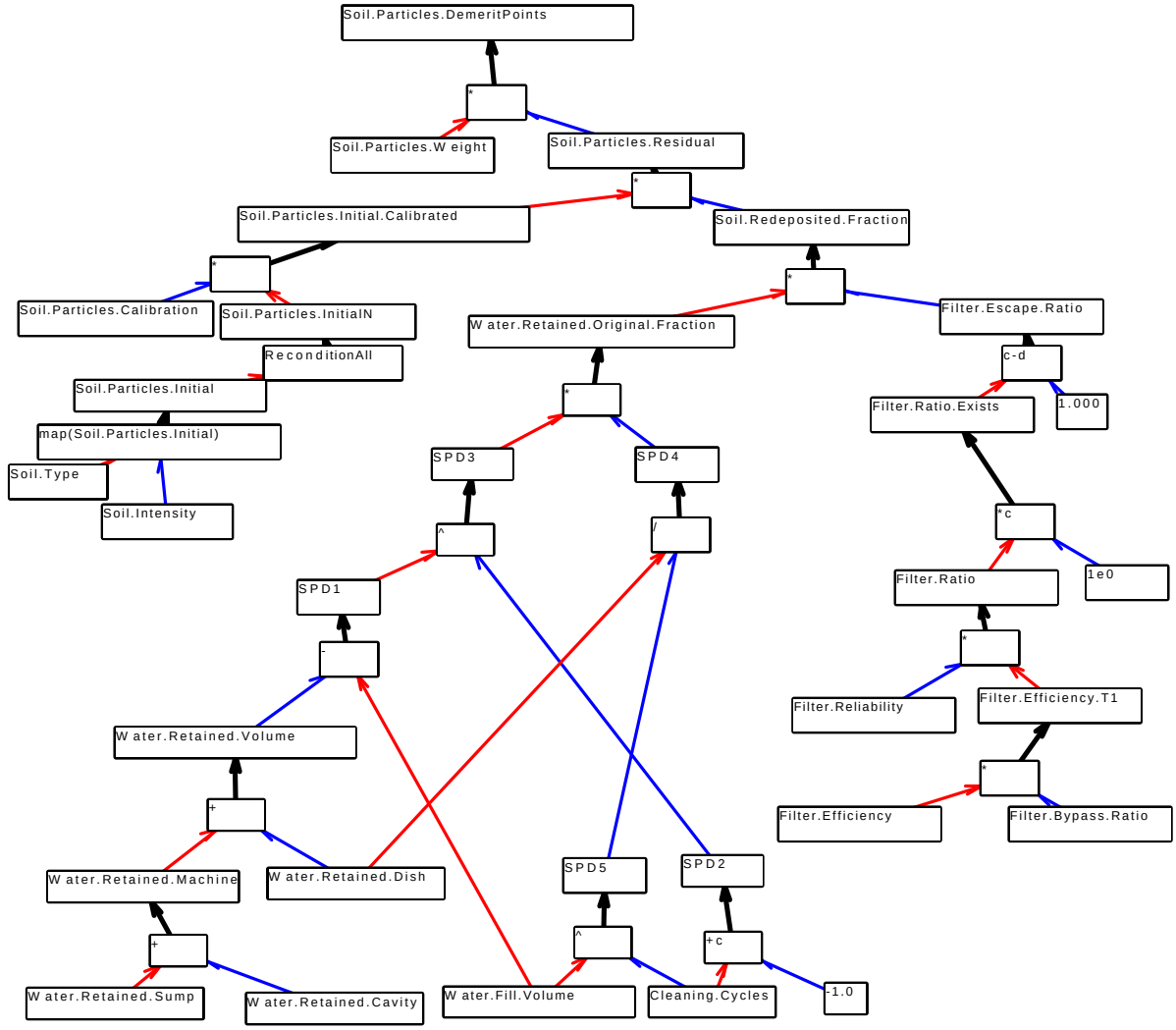


Figure 12.23: Computation graph for the removal of soil particles by rinse and filtration.

12.3.2 Explanation of probabilistic model

Next the various aspects of this model are discussed, starting with filtration at the lower right.

Filter parameters

Filtration efficiency was set relatively low, for machines that only use a filter plate. Filter bypass ratio (misnomer) was set high as most of the flow is filtered when using a filter plate. Filter reliability was set relatively high as large holed filter plates do not block easily, and do not depend on valves and other active components. The filter escape ratio is then determined using probabilistic arithmetic and the result is shown in Figure 12.24. A constant of 1 or 0 may be used to switch the filter on or off in the model, with output 'Filter ratio exists'. Finally the fraction of soil that escapes the filter and is available for redeposition is one less than the filter ratio and is given by Filter Escape Ratio.

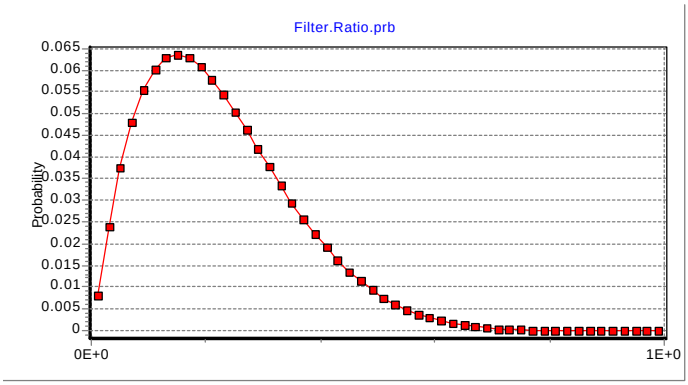


Figure 12.24: Filter ratio for the model.

Water retained volume

Water retained volume was determined as the sum of water retained in the sump, on the walls of the cavity, and on the dishware. Water in the sump varies according to the geometry of the fluid circuit and some machines with which the author is familiar show sump retention of 250 ml or even greater.

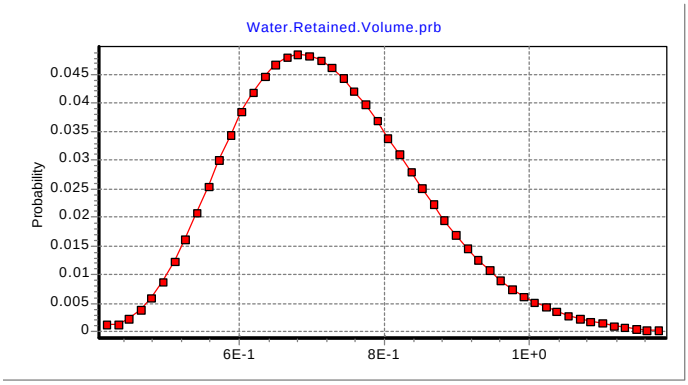


Figure 12.25: Total Water retained volume.

Wet items of dishware were weighed to determine water film retention so that the estimates could be informed by data, but the cavity film parameter is an uncertain estimate. Total water retained volume is shown in Figure 12.25.

Water fill volume

Large fill volumes are beneficial for wash performance. They dilute the suspended soil and thereby reduce the redeposition problem. Unfortunately large fill volumes also waste more water, raising the issue of social conscience. The operating cost to the user is also affected. However water usage is not reported on the front of a dishwasher, so it might not be a major consideration in design if it were not for its effect on energy consumption. Large fill volumes require large energy input to heat up to temperature, and energy consumption must be stated on the front of a dishwasher (at least in the USA market). The values used are shown in Figure 12.26.

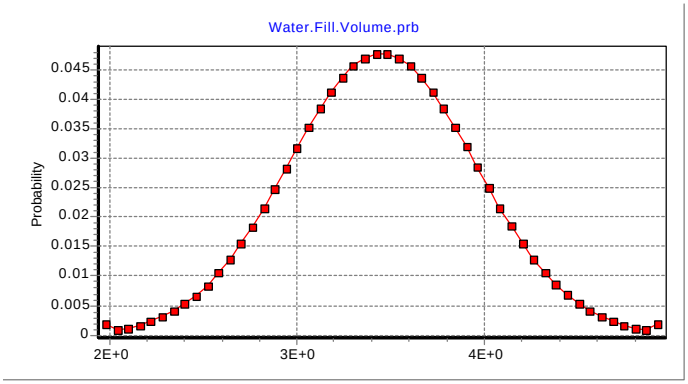


Figure 12.26: Water fill volume (per fill).

Water retained original fraction

The dilution principle ensures that the soil in the original cycle is diluted with each subsequent wash and rinse cycle. The mathematics are computed below this item in the model, and the results are shown in Figure 12.27.

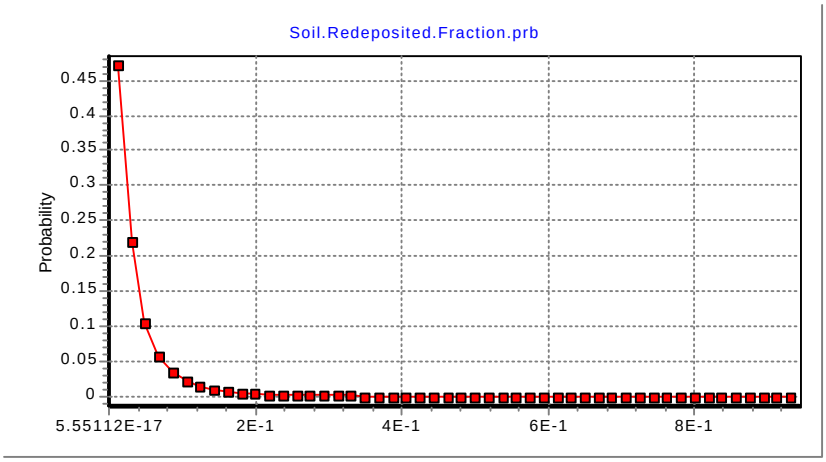


Figure 12.27: Water retained original fraction.

Note that ‘Cleaning cycles’ and other parameters provide inputs to two parts of the model. This is a legitimate arrangement in the Design for System Integrity (DSI) method, whereas special care would be required with this type of relationship if Monte Carlo was used. The reason is that DSI propagates the entire distribution (values and their probabilities) whereas Monte Carlo only takes a random value and has no means of knowing its probability until many more runs are complete. Monte Carlo analysts therefore have to ensure that they use the same random value where ever it appears in the model.

The product of ‘Water retained original fraction’ and ‘Filter escape ratio’ gives the Soil redeposited fraction, which is shown in Figure 12.28.

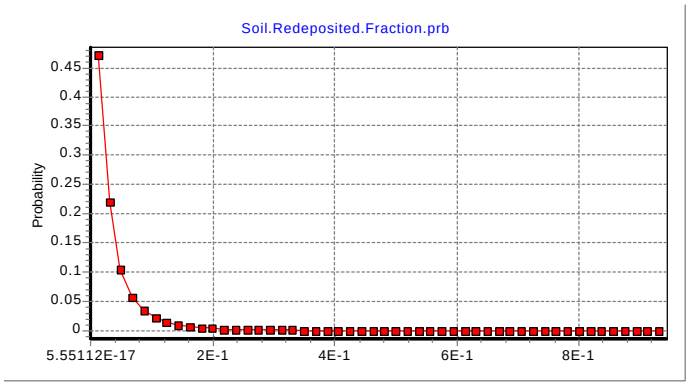


Figure 12.28: Soil redeposited fraction.

Soil particles initial

This parameter is a mapping between soil type and soil intensity. The soil type is the ANSI/AHAM profile described earlier, and the soil intensity provides for light, medium, or heavy with all the probability given to medium in this case. The ANSI/AHAM wash test does not prescribe soil intensity (or thickness) very closely. This possibly reflects an underlying belief that wash performance is primarily about getting soil off the dishware and that soil thickness plays an insignificant part in this process. In other words that thick soil is no more difficult to remove than thin soil, providing that the wash fluid reaches the soiled area. The test is then primarily to see whether wash fluid reaches the soiled area. However the interpretation followed here is that dishwashers usually have no difficulty in removing loose soils or soils that are soluble (or break down) in washing fluid. Dishwashers also remove coarse and fine granular soil and food fragments, though they tend to redeposit these. It is suggested that redeposition is a significant spoiler of wash performance for many dishwashers.

The map is used to determine the initial number of soil particles (soil particles initial), and it allocates higher counts for soils such as mashed potato that are known to challenge rinse effectiveness. The results are shown in Figure 12.29.

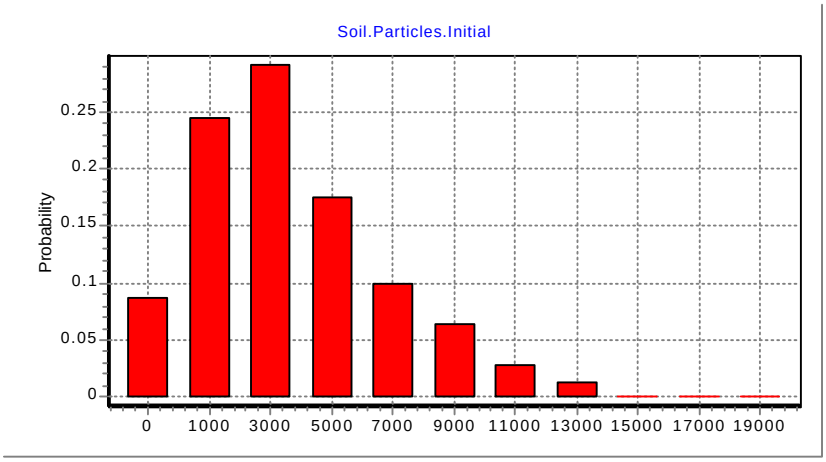


Figure 12.29: Soil particles initial.

The result was processed through the ‘ReconditionAll’ function to ensure the numerical scale was robust and then multiplied by a calibration factor (‘Soil particles calibration’ which is itself a probability distribution).

Soil particles demerit points

Soil particles residual is the product of the number of soil particles (‘Soil particles initial calibrated’) and the Soil redeposited fraction. It simulates the number of particles left on an item of dishware after all the wash cycles. It is multiplied with the demerit score per particle (‘Soil particles weight’) to give the total demerit points for particles on an item of dishware (‘Soil particles demerit points’), which is shown in Figure 12.30.

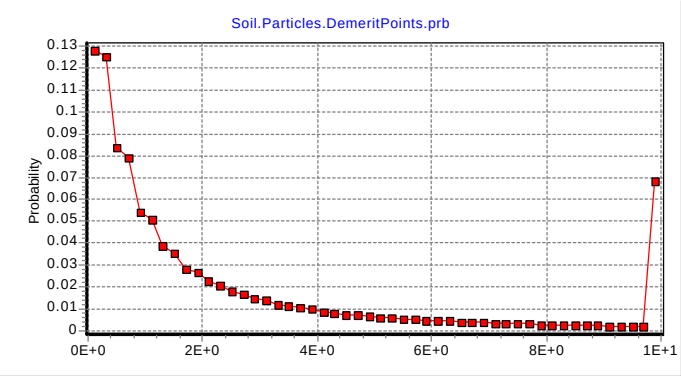


Figure 12.30: Total demerit points for particles on an item of dishware.

12.4 **Wash performance results**

The top level of the wash model was shown in Figure 12.1 and we now return to it as all the underlying parameters have been discussed. The underlying parameters were the demerit points per dishware item for each of un-removed patches of soil and redeposited particles. These are added together using a probabilistic addition to give Soil demerit points total as shown in Figure 12.31. '*Soil demerit points total*' is the thick curve. This is the probabilistic addition of the two lighter curves, the *'Soil Patch Demerit points'* (light curve with initial probability of zero) and the *'Soil particles demerit points'* (light curve with initial probability of 0.12)

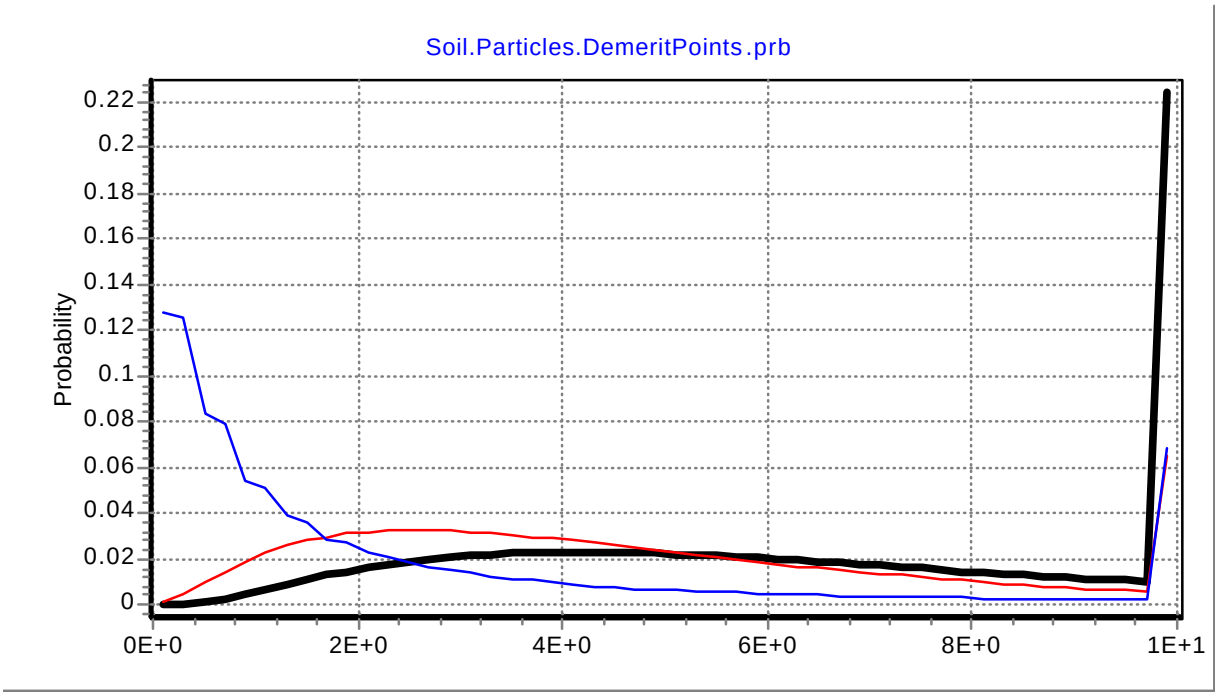


Figure 12.31: Total soil demerit points for particles on an item of dishware.

A probabilistic addition was used as the two inputs are independent. A case could be made for using a discrete addition, which gives the results shown in Figure 12.32, but this is not favoured as (i) the problem is not one of trying to find an average, and (ii) the discrete factor (0.78 weighting was used for the light curve with initial probability of zero) does not readily have a meaningful interpretation in the model. The discrete addition approach was therefore discarded.

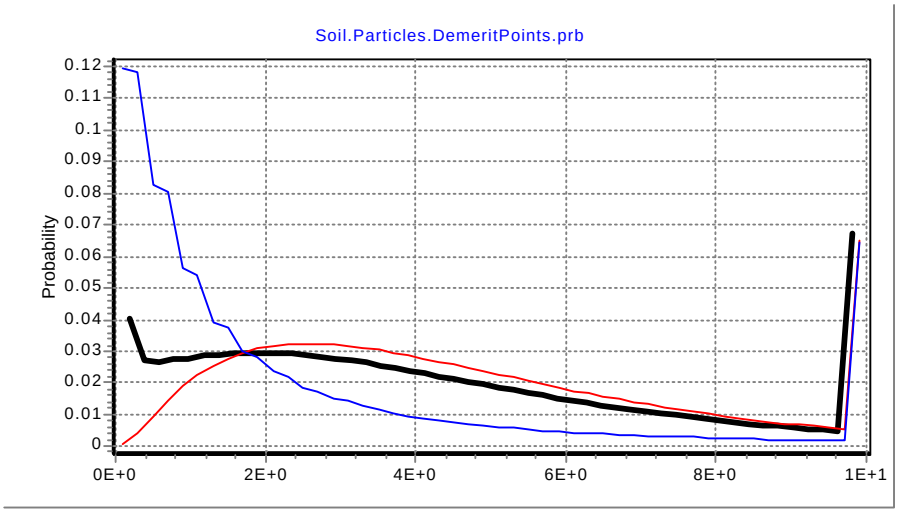


Figure 12.32: Discrete addition of two input curves (light) to produce an output (thick curve).

The rest of the calculation for wash performance is straightforward. The total demerit points are truncated at maximum 10 (using the 'RetainBelow' function), converted to a merit based wash score per plate, and then multiplied by 10 to give a percentage. The final result is shown in Figure 12.33, being a duplication of the figure in the previous chapter.

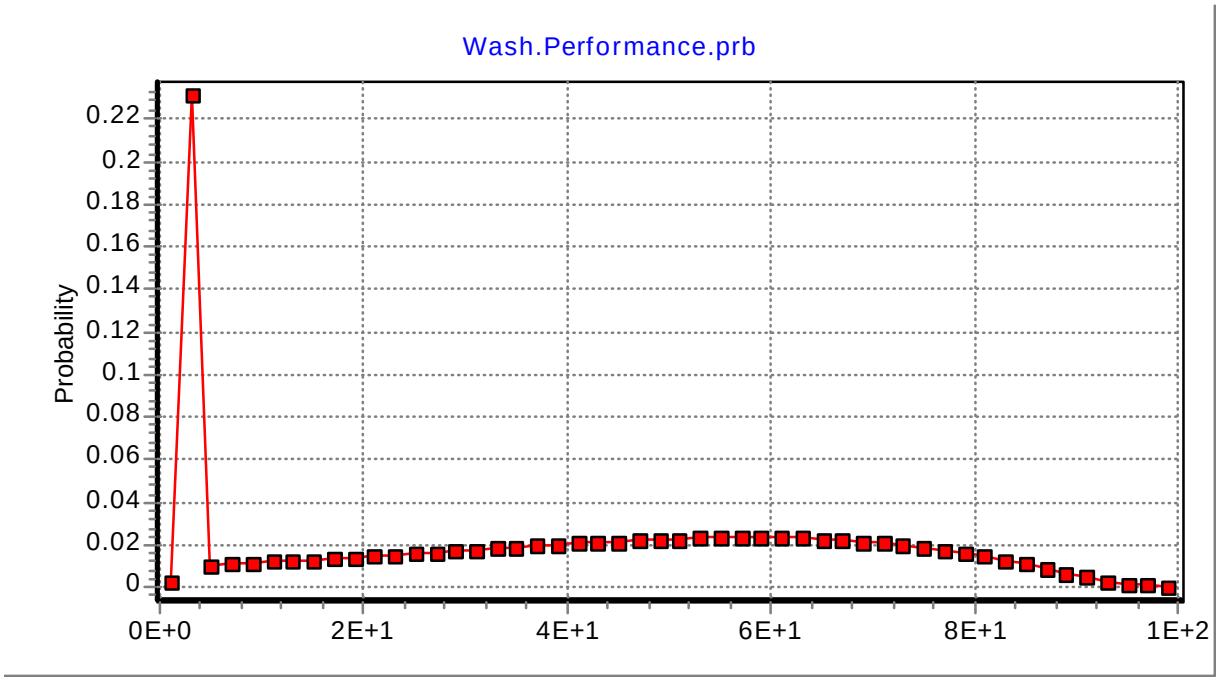


Figure 12.33: Wash performance [%] predicted for this set of model parameters.

12.5 **Conclusions**

A new model for simulating wash performance of domestic dishwashers has been proposed. The model incorporates two primary effects, removing the soil from the dishware, and preventing it from being redeposited. In turn these are driven by a large number of input parameters including but not limited to soil type, wash time, wash temperature, sprayer type, fill volume and filter properties.

The model accommodates large uncertainties which it represents as probability distributions, and propagates them through the system using probabilistic computation. Expert opinion is readily accommodated and even essential, as there are large parts of the model that are qualitative in terms of the data and qualitative in terms of the relationships too. This chapter has set out the principle parameters of the model with motivations where appropriate, so that others can build on it or provide alternative interpretations. Potential use of the model could be to explore configuration issues at early design.

Chapter 13

Simulation of dishwasher hydraulics at early design

The hydraulic performance of a dishwasher fluid circuit is simulated at the early design stage where uncertainty is high. The approach was to use probability distributions to model uncertainty in parameters, and then use a probabilistic computation engine, the Design for System Integrity methodology, to propagate the uncertainty through to the output parameters. Results from the probabilistic method are compared to those obtained by a deterministic approach.

The probabilistic approach yields greater information quality for decision making since it makes it possible to assess the risk and uncertainty in design parameters. The method shows that it is possible to extend reasoning tools into the early design stages where uncertainty is high.

13.1 **Introduction**

The problem studied here is to determine the hydraulic performance of a pump and spray system for a dishwasher. In particular there is a need to calculate the jet velocity for the water that strikes the dishware. This affects the performance of the dishwashing activity, as well as the noise produced by the machine, via qualitative mechanisms that are not further considered here. The challenge is to determine the jet velocity early in the design process, while the input parameters are uncertain. These input parameters include the characteristics of the wash pump, the layout of the plumbing, and the nozzle configuration. This chapter describes the application of the Design for System Integrity (DSI) methodology to this problem.

13.2 **Deterministic approach**

The conventional approach to determining the performance of a system is to apply known principles, in this case those of fluid mechanics, to calculate an answer. The difficulty is that often only part of the required information is known, a problem that is especially prevalent at the early design stages. It is therefore necessary for the designer to estimate or guess those parameters which are unknown or undecided. This is a deterministic approach in that a single value is used for each parameter, and only one final value (that of jet velocity in this case) is produced. There is no indication as to how likely that final value is other than the hope that if reasonable values were selected for all the inputs, then the final value should also be reasonable. Even so, there is no indication as to the possible spread in the final value. These limitations can be overcome by using a probabilistic approach, but before we do so it is necessary to develop the deterministic model.

The principles of fluid mechanics may be applied to determine the deterministic equation for jet velocity. It may be shown that the jet velocity V_{jet} is

$$V_{jet} = \frac{Q}{A_{nozzle}.N_{jet}.C_c}$$

where

Q discharge [m^3/sec]

A_{nozzle} area of one nozzle [m^2]

$A_{nozzle} = pi/4 . D_{nozzle}^2$

assuming all nozzles are the same area

D_{nozzle} diameter of nozzle [m], eg 0.002 m

N_{nozzle} number of nozzles, eg 20

C_c nozzle coefficient of contraction, eg 0.65

The difficult parameter to determine is the discharge, especially if the characteristic curve of the wash pump is unknown. It may be shown that this is approximately given by:

$$Q = \sqrt{\frac{(H_{mano} - H_{lift}).2.g}{\frac{k_{bend}.N_{bend} + k_{nozzle}}{(\frac{\pi}{4}.D_{pipe}^2)^2} + (\frac{1}{C_v.C_c.N_{jet}.A_{nozzle}})^2}}$$

where

H_{mano} manometric head of wash pump, given by

$$H_{mano} = \frac{1}{g}.(\omega.\frac{D_{impeller}}{2})^2$$

which assumes a centrifugal type pump and radial input flow

g acceleration due to gravity [9.8 $m.s^{-2}$]

w rotation speed of wash pump impeller [rad/sec] eg 1500 rpm

$D_{impeller}$ diameter of wash pump impeller [m] eg 0.050m

H_{lift} static lift height from wash pump to nozzles [m], assuming this is one fixed height, eg 0.500 m

k_{bend} shock factor for plumbing system, assuming that this includes shock and friction, eg 0.45

N_{bend} number of bends in plumbing system, assuming the bends are all the same, eg 5

k_{nozzle} shock factor for nozzle, given by $k_{nozzle} = (1/C_v^2 - 1).N_{nozzle}$

C_v nozzle velocity coefficient, eg 0.98

The appropriateness of the model requires some thought. There are a number of significant assumptions even in this deterministic model, and these will affect the final result. If the system were more completely defined, then it would be possible to produce a more accurate model. It would be particularly useful to have detailed knowledge of the characteristic curve of the pump, and of the configuration of the pipes and nozzles. With this information a more valid model could be produced. However having this type of detailed system definition means that the design would no longer be in the early stages. For early design it is appropriate to select a model that describes the performance even if imperfectly.

With the deterministic model developed, a designer may put in candidate values for the parameters, and calculate the output (jet velocity). Given the example configuration shown with the parameters, it is possible to calculate the system performance (eg V_{jet} = 4.39 m/s).

As with any deterministic analysis, the example values represent only one possible configuration, and there is no indication of the uncertainty inherent in the parameters. There is the risk that the designer may have selected input values which are biassed. A partially solution to this is to vary the input parameters and see how the output is affect, often called a “what-if” analysis. The method may be extended to *sensitivity analysis* in which is determined the extent to which a change in a parameter effects on the output. However these methods only partially indicate the uncertainty of the

outcome. For better analysis of risk it is necessary to use a more powerful probabilistic computation approach.

13.3 **Probabilistic approach**

Each of the parameters in an analysis may take a variety of values. The early design stages introduce a large amount of such uncertainty, because the design decisions are incomplete. For example the designer might not have selected the wash pump, and therefore characteristics such as impeller diameter and speed cannot be given in a deterministic sense. As the design process firms up, so the uncertainties narrow and in the limit they converge to the manufacturing process variability of that parameter[161].

To implement the DSI method it is necessary to set up a sequence of operations. This is represented by the graph shown in Figure 13.1. This is nothing more than a graphical representation of the equation for the deterministic case, though it includes intermediate calculations that are not explicit in the equation.

---

[161]The conventional approach to analysing the uncertainty is using *Monte Carlo* analysis. This takes random samples from each probability distribution, and finds the result. The process is repeated many times to build up the output distribution. It is a powerful method in that it is able to accommodate any probability distribution. It has been used in a variety of domains (Kleijnen, 1995; Zhang et al, 1992: Duckworth et al, 1998; Basu, 1998; Kostetsky, 1994).

The DSI method applied here has a different underlying mechanism to Monte Carlo, though it produces comparable results. The main difference is that the DSI method uses a complete probability distribution, which it processes in a combinatorial fashion with other distributions, whereas the Monte Carlo method takes single random samples from the input distributions and computes the output. The DSI method also has the ability to process qualitative relationships (though not demonstrated here), a feature which is not possible with Monte Carlo simulation. The uncertainty about convergence, an essential issue of Monte Carlo simulation, is also avoided. Accuracy is instead determined by the fineness of the discrete input distributions.

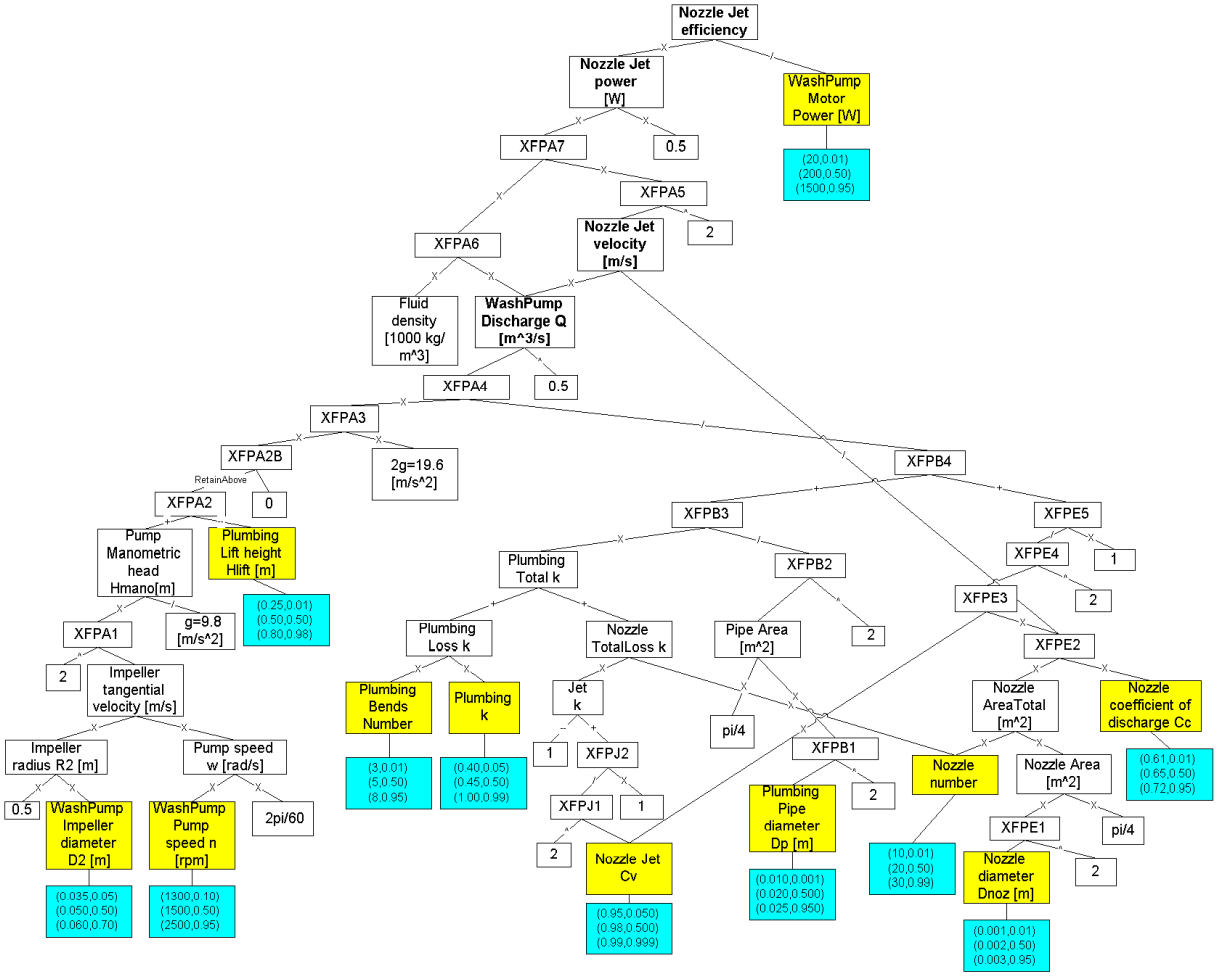


Figure 13.1: Graph of relationships used to solve the probabilistic reasoning aspects of dishwasher hydraulic performance. The graph shows the computation steps necessary to determine the top parameters. Shaded blocks show parameters that are primary assertions, that is information that the user has to provide. Some sample data are shown as estimate pairs (value, cumulative probability). Unshaded blocks are intermediate values that result from calculations on the assertions and the constants. In some cases an intermediate parameter has a specific interpretation, in which case it is labelled as such, eg area is determined from pipe diameter and some constants. However many other intermediate values have no significant physical interpretation, and are labelled starting with 'X'.

The operators used in the calculation are shown in the arcs of the graph. In some cases the second parameter is a deterministic value, for example radius is diameter/2, in which the '2' is known exactly. In such cases the uncertainty about the parameter '2' collapses. Otherwise the values are given by a probability distribution.

The DSI method provides several mechanisms by which probability distributions may be specified. The mechanism used here is for the user to specify estimates. These might be the highest, lowest, and typical values, together with the their probabilities of occurrence. For example, WashPump Motor Power [W] might be estimated at the early design stages as 1% chance of being less than 20 W, 50% chance of being less than 200 W, and 95% chance of being less than 1500 W. This translates to the estimate pairs (20,0.01) (200,0.50) (1500,0.95) which are evident in the figure. In this model there are three estimates for each asserted distribution, though the method permits more estimates if necessary.

To ask a design expert to specify an uncertain parameter in terms of three estimates is not unreasonable, even if there is considerable uncertainty in the parameter. The greater the uncertainty the greater the spread in the parameter. Use of three parameters is consistent with the three parameter estimates to model project management task duration in *PERT* analysis. The difference between the DSI and the PERT methods is that the former fits a distribution to the estimates and propagates that through the system, whereas PERT simply uses the three estimates as-is using methods that only approximate to full probabilistic computation.

The DSI method takes the estimates, and fits a Normal distribution (other types are also supported), and then propagates that distribution through the model.

13.4 **Results of probabilistic computation**

The result of the probabilistic approach is a probability distribution for the parameter of interest. Each parameter in the model, including the intermediate values is a probability distribution (excepting constants). The method saves each distribution as a file for later inspection. It also accepts alarm settings from the user, and will alert the user if the alarm conditions are violated. At the end of the processing, the user may inspect the final and all the intermediate probability distributions.

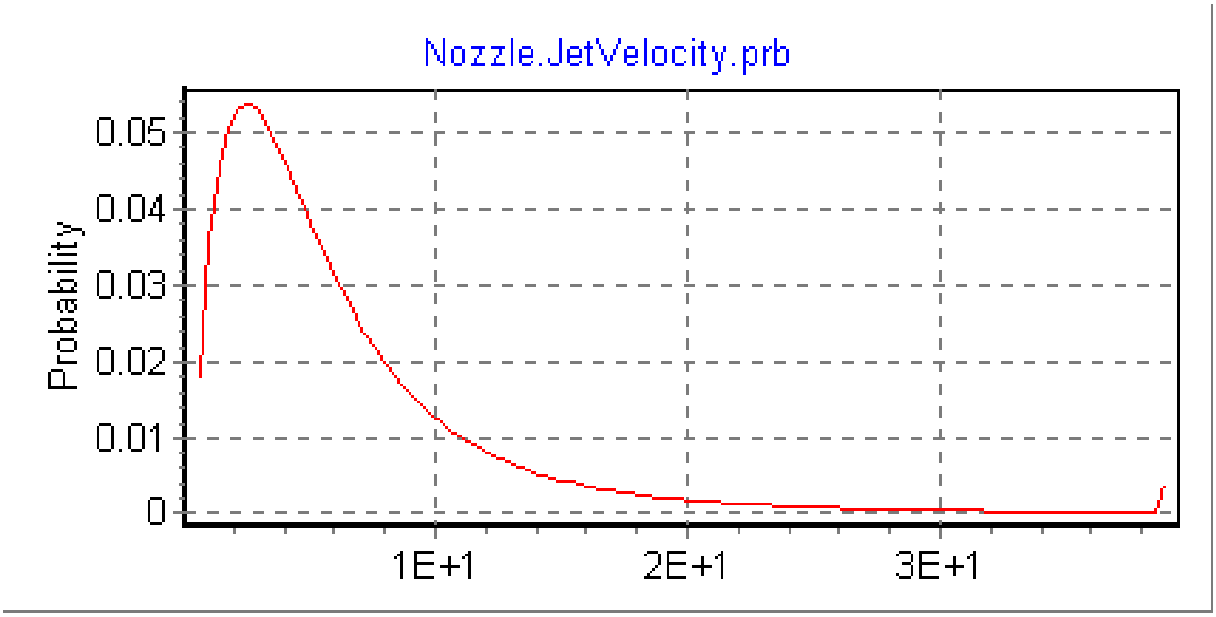


Figure 13.2: Final result for Nozzle Jet velocity, after probabilistic computation. This is a probability density, which should be interpreted as a histogram. The horizontal axis is jet velocity [m/s] and the vertical axis is probability density. The shape of the distribution shows a peak at 2.6 m/s which is the mode. There is no lower tail to the distribution below 0 m/s, and this is consistent with the physical interpretation that jet velocity may not be negative. The distribution does however have a long upper tail, so that velocities in the region of 20 m/s are possible, but with low probability. Neither the mean nor the median are apparent from inspection of this chart. The value of this chart is that it shows the shape of the distribution and where the results are concentrated.

The results appear as density (also called frequency) as well as cumulative distributions. The density distribution shows how the results are distributed, similar to a histogram. The final result is shown in Figure 13.2, which also includes a discussion on interpretation of the density distribution. The mode = 2.6 m/s is easily determined from the density distribution, simply being that value of velocity where the density peaks. However this is not a reliable indicator of the typical performance. This distribution has a shape that is very different to the Normal as it is asymmetrical and bounded on one side. This is not entirely unexpected, as there are many product operations in the equation, and these tend to produce an Exponential shaped distribution (Vose, 1996).

For greater depth of quantitative understanding of the variability, the cumulative distribution (Figure 13.3) is more valuable. It is closely related to the frequency distribution as it is simply the area under the frequency distribution to the left of the point of interest. The cumulative distribution gives the user the means to quantify risk. For example the probability of getting a jet velocity lower than 2m/s may be determined graphically as 17%, which is not apparent from inspection of the density. Percentiles including the median = 4.6 m/s may be determined by inspection of the cumulative distribution. The median is that value with 50% probability of occurrence, so that half the time the result would fall below it and half the time above. The mean = 6.4 m/s may also be determined, but this requires calculation by the software rather than inspection. The mode, median and mean are all different for skewed distributions like this. The mean is a weighted average, so that more extreme velocities such as 20 m/s contribute proportionally more to the mean.

In this case the jet velocity was calculated as 4.4 m/s with the deterministic method. The median as calculated using the probabilistic approach is close to this, and is certainly much closer than the mean. The difference between the deterministic result and the median appears to be due to the closeness of fit (or lack thereof) between the user’s three-point estimates and the fitted distribution. The deterministic result was determined solely on the middle value for each input variable, ignoring the lower and upper estimates.

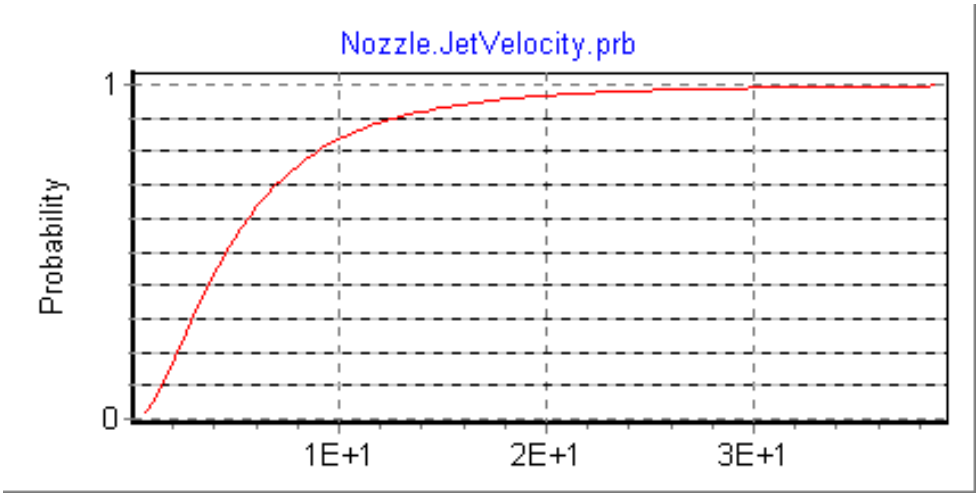


Figure 13.3: Cumulative probability for Nozzle Jet velocity. This is a companion chart to the density shown above. The horizontal axis is jet velocity [m/s] and the vertical axis is the cumulative probability (area under the density curve up to and on the left of the point of interest). Cumulative distributions usually have an "S" shape, which this one displays to some extent. The cumulative distribution allows risk to be assessed. For example the probability of a velocity less than 2 m/s may be determined by inspection as 17%. Likewise the probability of a velocity greater than 20 m/s may be read off the chart as (1-0.97)=0.03 = 3%. Another important statistic is the median, which is the velocity for 50% probability, and this is 4.6 m/s by inspection. Half the time the velocity could be expected to be below this value, and half the time above it. Other percentiles may also be determined from the chart. However the mean is not apparent from inspection of this chart. The value of this chart is that it shows the risk of various outcomes, and permits those risks to be quantified.

13.5 **Conclusions**

The application of probabilistic reasoning to the early design stages has been illustrated with reference to simulating the hydraulic performance of a dishwasher fluid circuit. A deterministic approach was illustrated, using single-point estimates and an approximate fluid analysis to calculate the desired output of jet velocity.

While a simple deterministic approach is adequate to give the typical results from the system, it cannot show what the range of probable outcomes is, and with what likelihood. Nor is there much value in computing worst and best case scenarios by using only the lower (upper) estimate of each parameter. Though easy to do, it gives

extremely conservative results. The reason is that it is extremely unlikely that all the parameters would simultaneously take the worst (best) case. Such worst and best case scenario analysis does not determine the probability of those scenarios.

The probabilistic approach also calculates these extreme values, but avoids the associated problem of interpretation by providing the probability along with the result. The probabilistic approach requires for input not single-point inputs but probability distributions. The software provides a number of mechanisms to define such distributions, of which the provision of three estimates (lower, typical and upper) was used here.

The end result of the probabilistic approach is a probability distribution for the jet velocity, as well as all intermediate parameters that were calculated. The provision of a distribution makes it possible to quantify the risk of various outcomes of interest. The probabilistic method produced a median that was close to that of the deterministic method.

The results show that the DSI methodology is able to provide probabilistic computation. Furthermore, a probabilistic approach yields greater information quality for decision making than a simple deterministic approach, since it makes it possible to assess the risk and uncertainty in design parameters. Against this must be balanced the extra effort required to generate a probabilistic model, and the greater computational time compared to a deterministic model. Inevitably the effort, though not excessive, will limit the application of probabilistic reasoning to those applications that have greater importance. More fundamentally, the method shows that it is possible to extend reasoning tools into the early design stages where uncertainty is high.

Chapter 14

Simulating dishwasher Reliability

This chapter describes a reliability model developed for dishwashers, using probabilistic computation methods to propagate failure distributions.

14.1 **Introduction**

Reliability is a measure of how well a product performs its intended function over time. This is an important factor in the technical and commercial success of all products. A lack of product reliability exposes the manufacturer to risks such as customer dissatisfaction, poor product differentiation in the market place, costs of servicing the warranty calls, and unsafe failure modes with product liability claims.

The term “reliability” has a specific as well as a general meaning. The specific meaning is the probability of product failure at a given time[162]. The similar term *availability* refers specifically to the probability of a repairable system being functional at a given time. ‘Reliability’ is also used in a general way to refer to quality type issues, for which the term *dependability* is sometimes also used.

The benefits of reliability engineering are firstly the minimisation of life cycle costs. Costs of assembly rework, service, customer ownership may be minimised (Punches, 1996). The implementation of reliability engineering starts at the design stage even if the effects are seen far downstream. Design changes should be evaluated for reliability, especially to see whether or not they introduce new problems (Burgess, 1987). Reliability also provides a tool for managing the development process, since it requires that engineering changes be adequately justified (Punches, 1996). Reports on reliability and corrective action provide a history of product performance and are a valuable resource for future product development (Burgess, 1987).

Reliability tests, in increasing order of thoroughness, are given by Burgess (1987) as:

- Component tests are performed early in the design process, to check whether critical components are suitable. However the results are unsuitable for

[162]Reliability is the probability that a component or system will perform adequately (will not fail) up to a given time. Reliability varies, depending on the given time. Systems have high reliability in the young ages, but reliability decreases with time. The reliability is not a single number but is always accompanied by the time concerned. Time should be interpreted in the broader sense to also include load cycles, missions and other measures of duration.

predicting the reliability of the whole machine since environmental conditions are seldom appropriate.

- System tests are done on complete machines, under conditions that closely match actual service.
- Reliability demonstration tests are used to prove to the customer that a certain level of reliability has been achieved.

The critical factors in product development are *performance*, *cost*, and *time*. Performance refers to how well the product meets the needs of the user, and is ultimately assessed by the user. For each product there is a primary function or design intent that it performs. There are always other performance criteria, one of which is reliability. Design engineers are generally successful in ensuring that the primary functionality is provided in a product. Therefore most products provide adequate function when they are new, otherwise they would not be on the market at all. Reliability measures the likelihood of that function still existing as the product ages. Naturally all products fail eventually by the processes of entropy, and users accept this as a fact, even if sometimes unwillingly. However it is a problem when the failure occurs too soon (in the user's eyes) after the machine is commissioned.

The cost issues include those costs necessary for product development, operation, maintenance, and disposal. These are often termed *life cycle* costs. Purchase and ownership costs are included. The timescale issues concern when the product will reach the market, and how long it will stay there. During product development these factors interact with each other and a critical function of *managing* the design process is to find a suitable balance between them. A deficiency in any one of performance, cost or time can substantially reduce the success of the product.

Global competitiveness, narrow windows of opportunity, and concurrent engineering place the product designer under intense time and cost pressure. Although performance is prominent in the minds of the designer, being the reason for creating the product and the rationale under which it will be sold, it sometimes happens that there are insufficient resources to adequately explore other viewpoints such as

reliability. Consequently it is not uncommon to find new products going through significant “teething problems” both in manufacture and in usage by the customer. Unfortunately this can lock the manufacturer into a process of having to make multiple small design changes, each of which has to be managed by way of change notices to production and field service staff. These changes are costly and disruptive. Production pressure means the changes are often done in a hurry, and it may be difficult to screen them beforehand for reliability implications. Consequently they may cause their own problems. This can cause a cycle of redesign activities, which distracts resources from other initiatives. Addressing reliability at early design requires the resources to anticipate and design against the potential failure modes. It is necessary to consider not only the failure of the primary design intent, but also of the other performance metrics for the product.

The balance between performance, cost and time is one which is made uniquely for each product development. For example, emphasising a performance criteria such as dependability may result in greater product purchase cost. However purchase cost is only one of the costs of ownership, and it may be that the improvements in operating and maintenance costs outweigh the increase in purchase cost. There is opportunity in the market place for a variety of products which differentiate themselves along such lines, though they otherwise provide the same primary functionality. Therefore the balance that a company selects between performance, cost and time is a critical element in the overall business strategy. To large extent the balance is affected by corporate vision, strategy and culture.

14.2 **Predicting Reliability at early design**

In developing a business strategy and managing the product development process, it is useful to be able to predict performance, cost and time factors early in the product development process, and especially to be able to conduct "what-if" analysis. The benefits are seen in being able to verify early on whether a design is likely to meet

reliability requirements, so that corrective action may be taken while the design is still flexible to accommodating change.

Other benefits of early reliability analysis are the determining of testing protocols. Prototype models cannot assess reliability adequately (O'Connor, 1981). If production is committed on the basis of prototypes only, then there is a risk of a flawed product entering service. It is useful to have a reporting system that informs the reliability engineers of new failure modes as they occur in the field. Quantifying the severity of the failure modes can be done by examining the warranty data that comes back from the field, though this can only occur once the product is already in service. This type of data is unavailable at early design, and instead it is necessary to supplement prototype results with predictive tools, of which the two main types are *failure mode and effect analysis* (*FMEA*) and *fault tree analysis* (*FTA*).

FMEA explores the possible failure modes for each device within the system. From physical devices it synthesises the failure modes. FMEA is used to define possible failure modes and identify which components require special design attention. It results in a table listing for each physical device the failure modes and the consequential effects on the bigger systems. Proposed solutions can be listed, and also probability of occurrence. Ashley (1993) motivates for the use of the related method of failure mode effect and criticality analysis (FMECA) as a technique to be used in product development. Another related method is hazard and operability analysis (*HAZOP*), which is widely used in many industries. An example would be Montague (1990) who describes how HAZOP methods systematically identify the ways in which process equipment can malfunction or be mis-operated (his application is the chemical engineering industry)[163].

---

[163]HAZOP is conducted by a team who have expert knowledge of how the plant operates. The team divides the process into sections such as vessels and other significant system, and applies the HAZOP to each section. The actions in the review are:

- define the intended function and process flows,
- describe the connections into the process (including utility services)
- determine ways in which each of the inputs to the process section could deviate from the intended value, by applying guide words
- examine possible causes for such deviation
- determine the consequences for such deviation
- suggest defences against these consequences

FTA works in the other direction: it starts with given failure modes of the whole system and explores into the system to find what device failure modes could be the root causes. Fault trees can easily show multiple failures by means of “and” nodes, whereas FMEA and HAZOP type methods are limited to single failures as they do not easily accommodate or even identify multiple failures.

Fault tree analysis is probably easier to use early in the design process, since it is easily interpreted (graphical), uses but does not require qualitative data, accommodates combinations of multiple failures, can represent human error, and forms the basis of diagnostic diagrams for service staff. Once the design has been firmed up then FMEA can be valuable, as it explores each device in turn for unanticipated failure modes, provides a means to quantitative failure (if required), and provides a table which makes it easy to assign responsibilities for corrective actions.

Reliability analysis at the early design stages is difficult. There are large uncertainties in product configuration: physical devices may not have been selected, or critical physical parameters may still be unresolved. While fault tree analysis can cope with this, FMEA may be more difficult to apply, and reliability of the whole product is even more difficult to predict. Test results at early design are sparse, and the tested prototype may not reflect current design thinking.

There are not many solutions, and to some extent the anticipation of reliability at early design seems to be considered by many as an intractable problem. Some use Bayesian methods as a solution, for example Mazzuchi and Soyer (1992) discuss the need to assess system reliability during development, and the difficulty of doing this with sparse test results. They motivate for the use of Bayesian methods to combine subjective information with available test data. This may be a useful method of incorporating prototype results and prior beliefs into reliability analysis. However the Bayesian methods are not without controversy because they allow subjective beliefs

---

- recommend actions to be implemented.

The output of the HAZOP study is a table listing the deviation, its causes, consequences and the defenses. A list of HAZOP guide words for chemical process is provided by Montague (1990).

to potentially contaminate hard data. Also, the single point estimates of reliability are of limited value as they do not show the uncertainty in the estimates.

14.3 **Creating a reliability model as a by-product of functional modelling**

The objective here is to assist the designer by creating a model of system reliability in the background, while the designer is working on other functional modelling aspects. Previous chapters have described the development of the Design for System Integrity (DSI) methodology, its probabilistic computation algorithms, and its ability to create other viewpoints in parallel to the one that the designer is working on. Placing a physical device (eg 'MyDevice') in any view, automatically creates an entry in the reliability view and assigns default reliability data to it, see Figure 14.1.

The designer may then put in the logical structure, for example to model devices that are in parallel or series. In the case of dishwasher and indeed many other consumer products there is no internal redundancy so all devices in the system are in series and the system fails when any device fails.

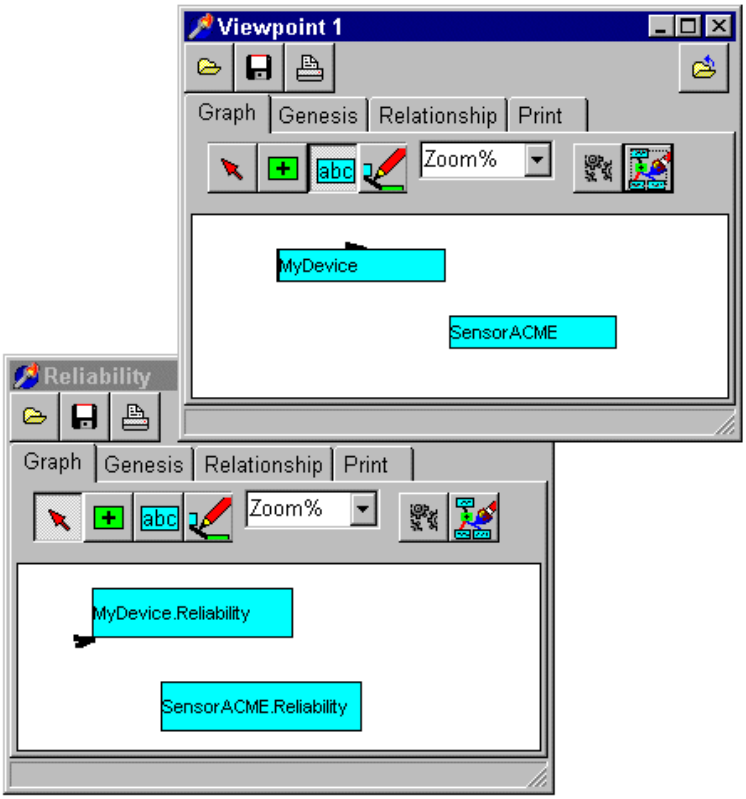


Figure 14.1: The DSI software will automatically create part of a reliability model (lower view) when relevant devices are used in another viewpoint (upper view).

The DSI provides strong graphing facilities, and therefore a tree structure may be created to compute the system reliability, as shown in Figure 14.2. It is necessary to create place holders for intermediate calculations, hence the arbitrarily named ‘Rx’ devices in the figure.

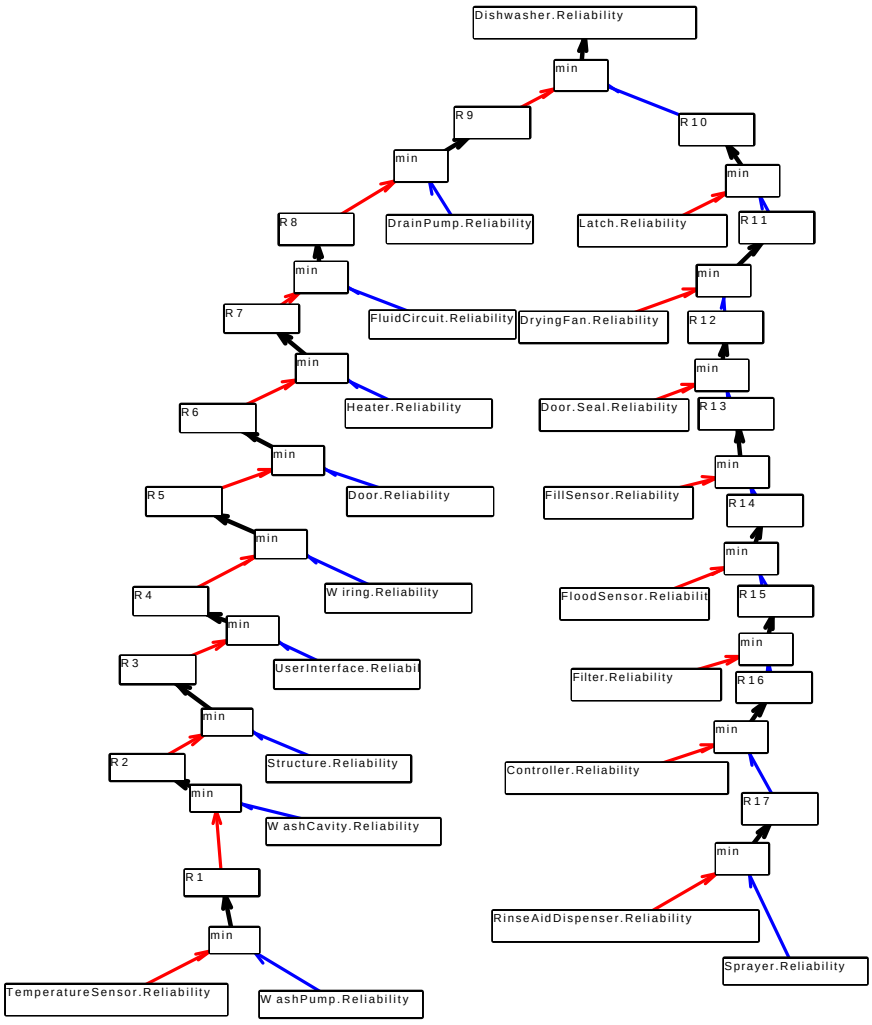


Figure 14.2: The designer can connect up the provided boxes with suitable relationships to determine system reliability.

Each of the blocks in the figure represents the reliability of a device, for example as a Weibull function giving reliability against time. This is consistent with data used for reliability analysis in general, and means that test data may be substituted where available. The designer may change any of the reliability parameters to reflect test

results or belief in a faster or slower failure process[164]. Many conventional reliability analyses such as fault tree analysis and Markov analysis require the use of Exponential failure distributions since they use mathematically explicit solution methods which only work on that distribution shape. Although the Exponential is indeed a valid distribution for many failure modes, especially of large systems, this constraint does not apply in the DSI methodology as it uses numerical solution methods. Consequently Weibull[165], Normal and other distributions may be used in the model, and the model does not need to be all one type either. The DSI software provides probabilistic computation with similar functionality to Monte Carlo simulation though using a different approach.

The model shown in the figure was based on representative but not necessarily accurate failure data. The failure data are default data extracted from a domain specific data file. When the user places a box (eg 'SensorACME') in the model, then the system prompts for the type of device (eg 'TemperatureSensor') and then retrieves the data for that type from the domain file. The designer can then edit those properties as necessary. By providing default data the system attempts to assist the designer who is at an early stage and does not have accurate reliability data yet, or does not know what reasonable reliability values are. If nothing else the system provides a prompt to the designer to think about the reliability viewpoint.

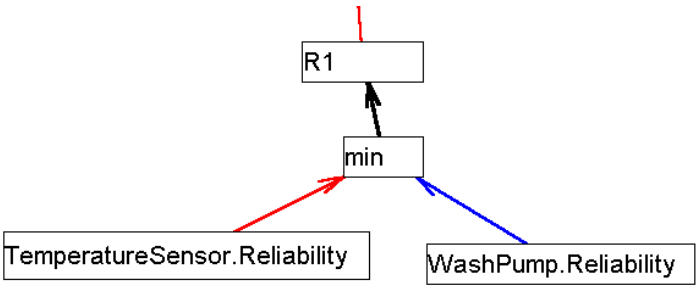


Figure 14.3: Graph using 'min' ('or') function.

[164]The DSI software does not currently provide explicit support for adjusting test results using Bayesian beliefs, but there is no reason why that computation should not occur outside of the software and the resulting distribution parameters entered in manually.

[165]A Weibull distribution with shape factor of 1 gives the Exponential distribution.

In computing the reliability of the whole machine it is appropriate to use the minimum function (‘min’) applied to the failure distributions of the constituent internal devices. In other words the device that fails first is the one that stops the whole machine. This may seem a very conservative approach, given that many devices will have distributions with some probability of failure immediately the machine is deployed. However it needs to be remembered that any one device has a relatively small probability of failing soon after the machine is deployed, and that probability is noted during the computation. For example, given the relationship shown in Figure 14.3, where two reliabilities are to be combined with the ‘min’ function, the result is as shown in Figure 14.4. The ‘min’ function is identical to ‘or’. Horizontal axis is time [yr] and vertical axis is histogram probability. One of the inputs has higher probability of failing early (the curve with the lowest peak and the longest tail) and therefore dominates the result at low time. However the other input has a sharp peak a little later, at which time it dominates the output. A machine made up of these two devices would have very little probability of getting into the high life regions, despite the one input having a long upper tail, because the other input would have caused failure before this.

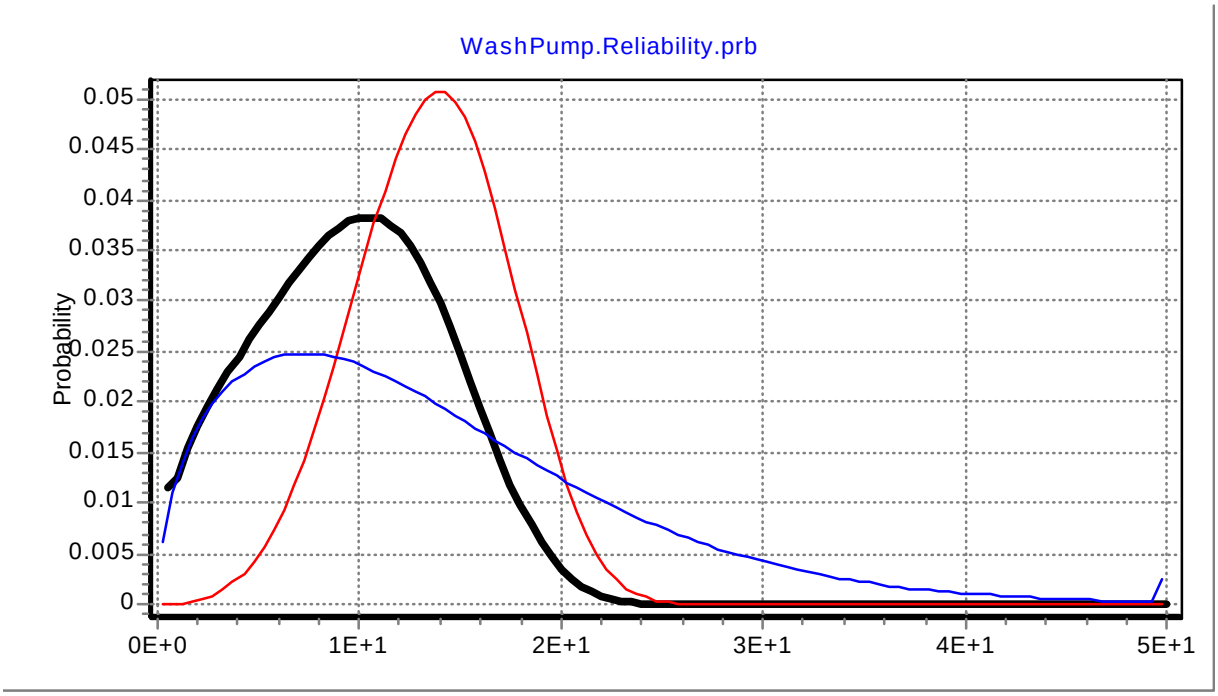


Figure 14.4: Result (heavy line) after combining two distributions (light lines) using the ‘min’ function.

On computing through the whole reliability tree, the result obtained is that of Figure 14.5. The data behind this are only representative of the genre, and so no special significance should be attached to the numerical values. This result should be interpreted as a histogram, with time [years] on the horizontal axis and probability on the vertical. The system shows high failure probability initially, decreasing with time. The shape is reminiscent of the Exponential distribution, even though few of the input distributions were of this type. However this is to be expected since it is well known that the reliability of a composite system tends to follow the Exponential distribution.

The same computation also provides the cumulative distribution, shown in Figure 14.6, from which percentiles may be found. The most important of these is the 50th percentile which is the median. Half the machine are expected to fail before this time.

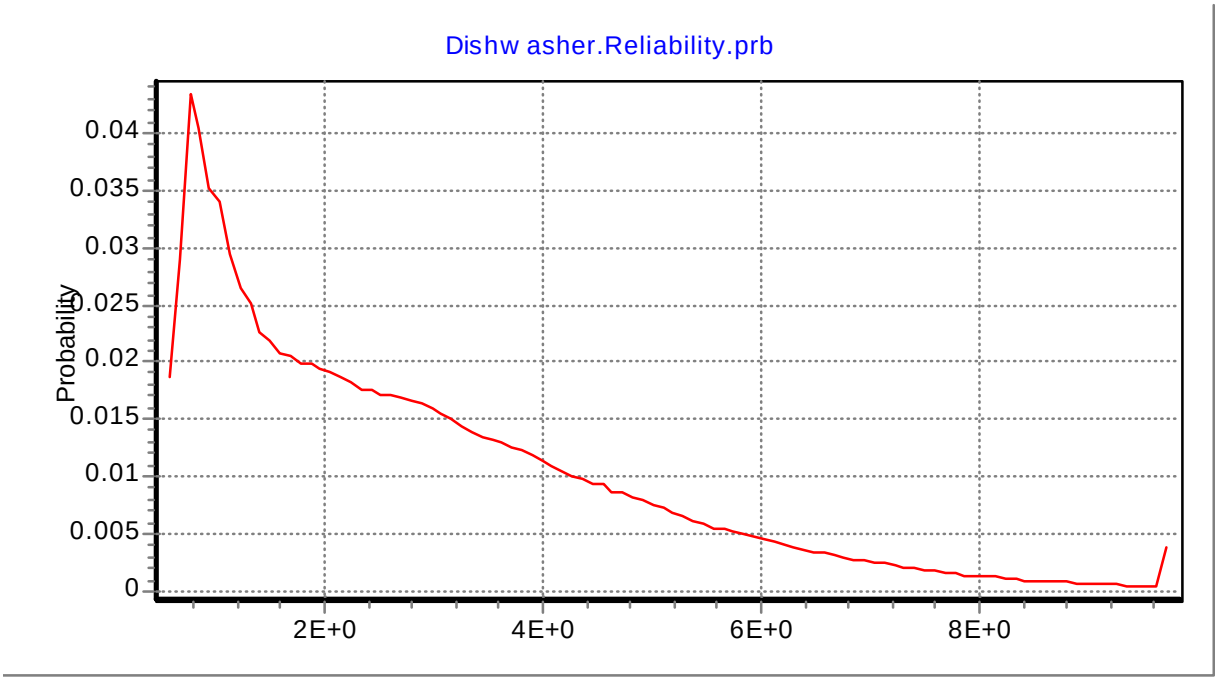


Figure 14.5: Failure probability of whole dishwasher, predicted on the basis of generic data for constituent device failure distributions.

Although most domestic appliances are notorious for failure, the data shown here are perhaps extreme. However this is deliberately conservative in an attempt to encourage the designer to collect such data as may permit a more generous interpretation. Reliability data may be obtained from prototype tests or data from similar products. Later, as the design moves into production, additional reliability data may be obtained from production reports, warranty claims and field service reports.

Most consumer products do not include redundant features, probably because the consumer market is very cost sensitive. However other design domains, such as aerospace and nuclear power, are safety and reliability sensitive and designers may include redundant components or even alternative systems (working on different operating principles) to improve the system reliability. The DSI software can readily model these relationships. In a reliability model the redundant components are connected with an *and* gate to show that both components must fail for the system to fail. The software accepts the *and* operator (it is identical to the *max* operator), and a model with this feature is shown in Figure 4.19.

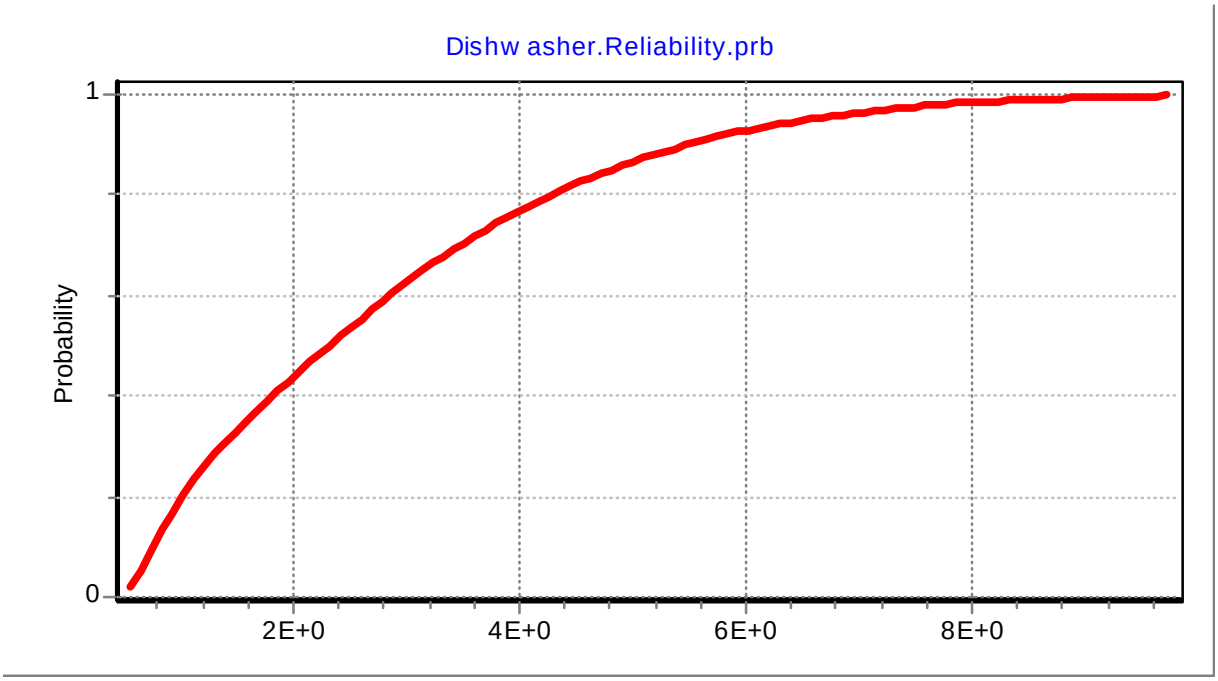


Figure 14.6: Cumulative failure distribution for generic dishwasher. Time in years is on the horizontal axis and cumulative probability on the vertical.

14.4 **Conclusions**

Reliability is an important viewpoint on the product. However manufacturers often find it difficult to incorporate reliability in the early design processes. In part this is due to the difficulty of applying the quantitative reliability methods during early design when information is sparse and prototypes are immature. In other cases manufacturers simply lack reliability-aware staff, or are unaware of how significantly reliability and warranty issues can affect financial profitability. Another problem in new product development is the application of patches to fix problems. In responding to a new problem (reliability or other) the designers create a solution and unwittingly introduce new failure modes. As the solution usually has to be implemented in a hurry because of market and production pressures, it may be impractical to thoroughly test it, and any new failure modes are only discovered when the revised product is in the market.

The DSI methodology provides a mechanism whereby reliability can be (i) modelled in parallel with other modelling tasks, and (ii) anticipated using generic data that are domain specific. Full probabilistic computation is supported, so that the uncertainty in estimates is explicit. The computation is not limited to Exponential distributions as are some other reliability tools. In these ways the methodology supports the early design process.

Chapter 15

Warranty risk for dishwashers

A probabilistic approach was taken to estimating reliability risk for dishwashers. A mechanism is provided for using beliefs, as expressed in confidence intervals to represent the uncertainty inherent in Weibull analysis. Consequently the method is able to be used at early design stages where reliability data are typically sparse. In this way the reliability viewpoint is able to be represented earlier in the concurrent engineering process. Application of probabilistic reasoning to reliability estimates provides a tool for the management of warranty risk and a means to direct design efforts to solve risky areas.

15.1 **Existing approaches to warranty**

Competition in the market ensures that most manufacturers provide a warranty for their product. This is usually in terms of a time period after purchase, and the manufacturer usually undertakes to repair or replace a product that fails during the warranty period, at no direct cost to the customer. The customer may experience indirect costs, chief of which is lack of availability of the product, but this is not covered by conventional warranties that apply in the domestic appliance industry.

Warranty cost can have significant financial impact on a manufacturer. The cost of the call-out, labour and replacement parts can be of the order of NZ$100 per warranty claim in the domestic appliance industry. New product development is particularly vulnerable in this regard, as new products tend to have new failure modes, some of which only become apparent when the product is in the field. The reliability approach to this problem is to conduct tests to establish the failure modes, and to quantify their severity.

However the reliability testing approach is difficult to implement at early design stages as design concepts are fluid, certain sub-systems may not have been designed, and prototypes are immature. Test results are often sparse, and those that do exist may be compromised by being done under conditions that do not match actual usage, or with prototypes that are no longer current, or with prototypes that do not embody manufacturing processes that are likely to be used later. Consequently manufacturers frequently leave reliability testing until product design is more mature, often as late as during pre-production. As the design and manufacturing process are committed at this time, there is limited opportunity to influence the design process. Unfortunately the management of the warranty process is sometimes only reactive: the warranty or quality control person waits for failures to occur in the field and then reacts to those by arranging corrective actions to the design.

There is a need for warranty prediction tools that can be used at early design. It is likely that an important competitive advantage may be secured by being pro-active towards warranty issues during the early design stages. The literature contains a limited amount of information on the determination of warranty issues. Greene (1989) computes the cost-effectiveness of a warranty from the viewpoint of the manufacturer. Isaacson et al (1991) quantify warranty risk with Monte Carlo simulation. Kostetsky (1994) discusses financial parameters such as cost and time to market, and describes the integration of risk analysis into the development process using Monte Carlo techniques. Hill et al (1996) developed a model for warranty cost, described as follows:

> *"The inputs to the model are the form of the life distribution (gamma, Weibull, or truncated normal) and its parameters, the lengths of the warranty period and of the life cycle of the item, whether or not the warranty renews upon failure of an item, the form of the rebate function, the seller's and buyer's costs and claim validation cost, the probabilities of a claim being made, validated, and of a new purchase being made, the form and rate of the discounting function, and the number of replications of the simulation desired. The output of the model are the mean and standard deviation of the n simulated costs generated for a given set of inputs. The model is validated by comparing its output against the true expected costs generated by a mathematical model in a few special situations where this is possible."*

The previous chapter has described the development of a system to anticipate machine reliability at early design. In this chapter the issue of estimating warranty risk is explored, and it is shown that a similar methodology can potentially assist the process.

15.2 **Predicting Warranty exposure**

In this chapter the term '*warranty exposure*' is used to refer to the uncertainty around the reliability of a product, so that reliability is given as a probabilistic parameter instead of a deterministic value. Practically all reliability and warranty methods are deterministic in that they predict a single value of reliability (eg reliability is 87% at time 2000 hr). The uncertainty or confidence interval around that reliability is seldom available. Thus the manager has no information on how good or bad the reliability could actually be. If reliability can be shown as a probability distribution then is it possible to gain this level of management. If there is a substantial probability of a poor reliability being obtained, then management can focus on how to improve the design and reduce the risks.

15.2.1 **Uncertain parameters for Weibull distribution**

The Weibull[166] distribution is widely used in reliability studies to model the failure probability of a device. It describes how the probability of failure changes with time. There is particular interest in the probability of failure at the end of warranty, i.e. at cycle time $t = T_w$. To quantify this, machines are usually put on test and the Weibull parameters determined by curve fitting to the experimental data. This conventional approach gives a single value for each of η, β and hence $R(T_w)$. The limitation with this approach is that there are uncertainties in the process. Firstly, the curve fitting

---

[166]The Weibull cumulative distribution for reliability is:

$$R(t) = \exp\left(-\left(\frac{t}{\eta}\right)^{\beta}\right)$$

where
R(t) reliability, that is probability of survival at operating cycle t
η characteristic life, a location parameter
β shape factor
The probability of failure at time t is simply F(t) = 1 - R(t) where
F(t) unreliability, that is probability of failure at operating cycle t

process has uncertainties (cf manual line fitting, least squares, maximum likelihood). Secondly there are different ways of processing the experimental data (cf ranking vs hazard analysis and others), especially when incomplete data are used. Incomplete data arise when not all devices have failed at the time the test is complete (or terminated), and some un-failed devices may still be at low cycles. Thirdly, it is common in the early design stages that the data are for a prototype that no longer represents the latest configuration. Fourthly, the test conditions are frequently an approximation to real usage conditions (cf accelerated testing).

Therefore there is a need to place a band of uncertainty around each of the Weibull parameters η and β. This is not easy to achieve in an objective manner. It is possible to address the first uncertainty, that of the scatter around the fitted line, and quantify it in terms of confidence intervals for η and β as per Mann and Fertig (1977) and Kinnison (1985). There does not appear to be any method that can provide a confidence interval for the second uncertainty (that of different methods). Finally there is the uncertainty that arises due to differing product configurations and test regimes. Others have used Bayesian probability for this type of problem. This provides a mechanism whereby if tests have been done to determine the reliability of a particular configuration (eg a prototype), then the subjective beliefs of an expert can be used to determine the reliability of another as yet unbuilt configuration. The method is not without considerable controversy due to the incorporation of subjectivity into a numerical process (Cox and Tait, 1993; Henley and Kumamoto, 1981).

In the current application all these uncertainties are rolled into one, and reliance is placed on expert judgement to propose the nature of the uncertainty around η and β. This is necessarily a subjective approach, but is justified on the grounds of seeking to extend reliability analysis into the early design stages where data is in any case sparse. Given appropriate methods and sufficient test time, it may be possible to determine objective estimates for the η and β distributions, but then the opportunity of influencing the early design stages is lost.

15.2.2 Warranty risk of domestic appliance

This study applies to a domestic appliance. Accelerated tests were done a collection of devices and Nelson hazard analysis used to determine η and β based on a least squares fit to the data. The devices were not of identical configuration, due to a variety of small design changes, but they were off a production line. A sample of the data is shown in Table 15.1, along with the hazard analysis equations. The resulting fitted curve is shown in Figure 15.1.

Cycles to Failure	Failed F	Available	Hazard Rate	Cumulative Hazard	Probability of Failure	TRANSFORMATION		Weibull cumulative
t	Censor C	s	z(t) = x/s	H(t) = sum z(t)	F(t) = 1-exp(-H(t))	x = ln(t)	y = ln(-ln(1-F(t)))	F(t) = 1 - exp(-(t/eta)^beta)
89	F	154	0.006	0.006	0.0065	4.488636	-5.036953	0.009856462
135	C	153	-	----	---			0.01678803
141	F	152	0.007	0.013	0.0130	4.94876	-4.337248	0.01774543

Table 15.1: Partial data set, with hazard analysis equation, for determining Weibull parameters.

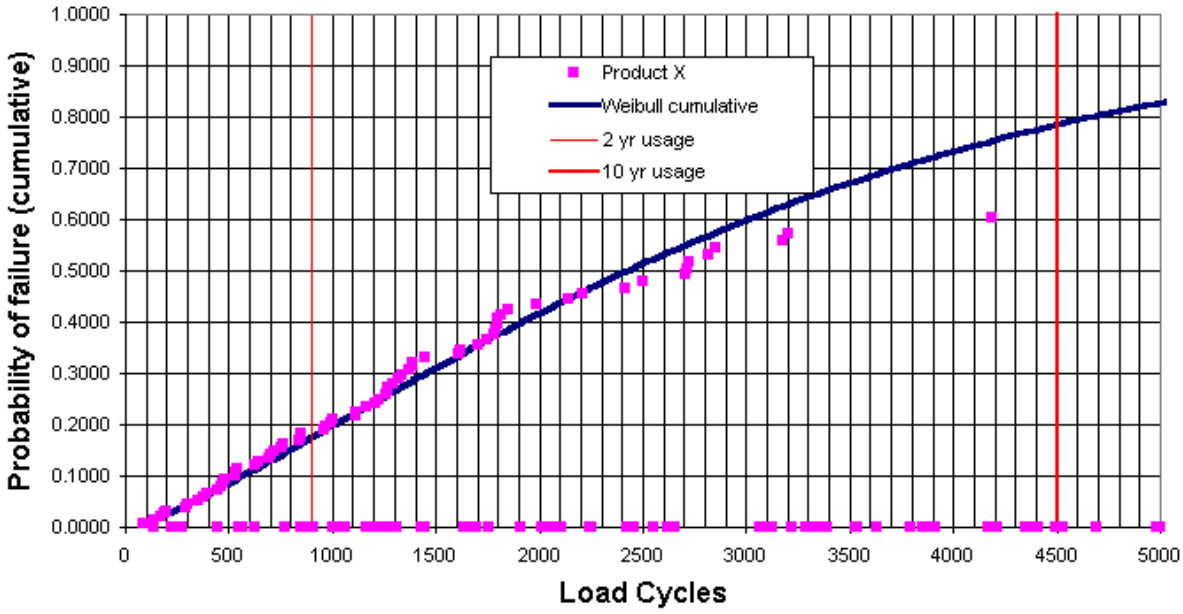


Figure 15.1: Weibull cumulative distribution fitted to experimental data. The horizontal axis is load cycles and the vertical is probability. The markers are data points and the heavy curve is the fitted Weibull curve.

The results were η = 3214 operating cycles and β = 1.287 for the Weibull distribution. With these results it is possible to predict the reliability at various times of interest. For example, if the warranty period corresponds to 900 load cycles, then the probability of failure during warranty is 18%. If the cost of servicing each warranty call was NZ$100 then the cost of this failure mode to the manufacturer is NZ$18 per machine (i.e across the entire production volume). If 100 000 of these devices are produced in a year, then the total warranty cost is NZ$1 800 000 per annum. These numbers are simply representative, but they indicate the potential magnitude of the warranty costs carried by domestic appliance manufacturers.

15.2.3 **Uncertainty bands around Weibull parameters**

Obtaining the above single point estimate of reliability is conventional reliability analysis. Next this is taken further by applying uncertainty bands around the two Weibull parameters. As discussed, this is a subjective process. However there are some clues that can be used, and if in doubt the estimates can be made wide to account for the uncertainty.

The value for shape factor β is the easier to estimate, since reliability lore suggests that it is often around 1.0 and seldom as high as 4.0. Therefore a suitable confidence interval may be β = (0.9 [10%], 1.287 [50%], 2 [90%]) where the percentages are the confidence limits, eg ‘10% of the time the value for β might be below 0.9'. The value obtained from the reliability analysis (1.287) is assigned to the median. A confidence interval for the characteristic life might be η = (2000 [10%], 3214 [50%], 5000 [90%]).

The next task is to fit a distribution to these estimates. There are several candidates, including the Normal, Weibull and Triangular. The Normal is often found to be unsuitable, because it is a symmetrical distribution and may fit the estimates poorly. The Triangular is a simple distribution but is unsuitable here as it cannot easily cope with the upper and lower tails implied in the estimates. The Weibull distribution is used here. The Design for System Integrity (DSI) software fits a curve to the three estimates and return the distribution parameters, β = Weibull(1.546, 3.819) and η = Weibull(3792, 3.374). It then does a probabilistic computation for the reliability. The model is illustrated in Figure 15.2.

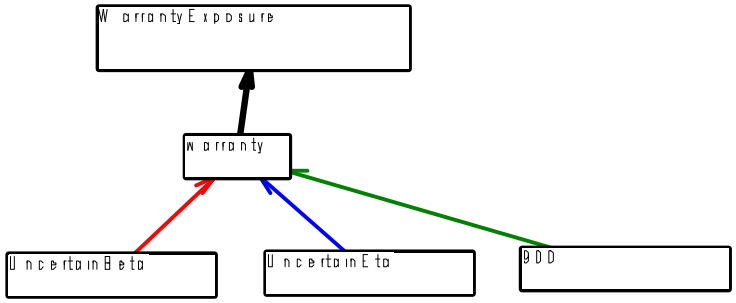


Figure 15.2: Model of warranty exposure with uncertain Weibull parameters. The function 'warranty' computes the Weibull equation using the uncertain input variables.

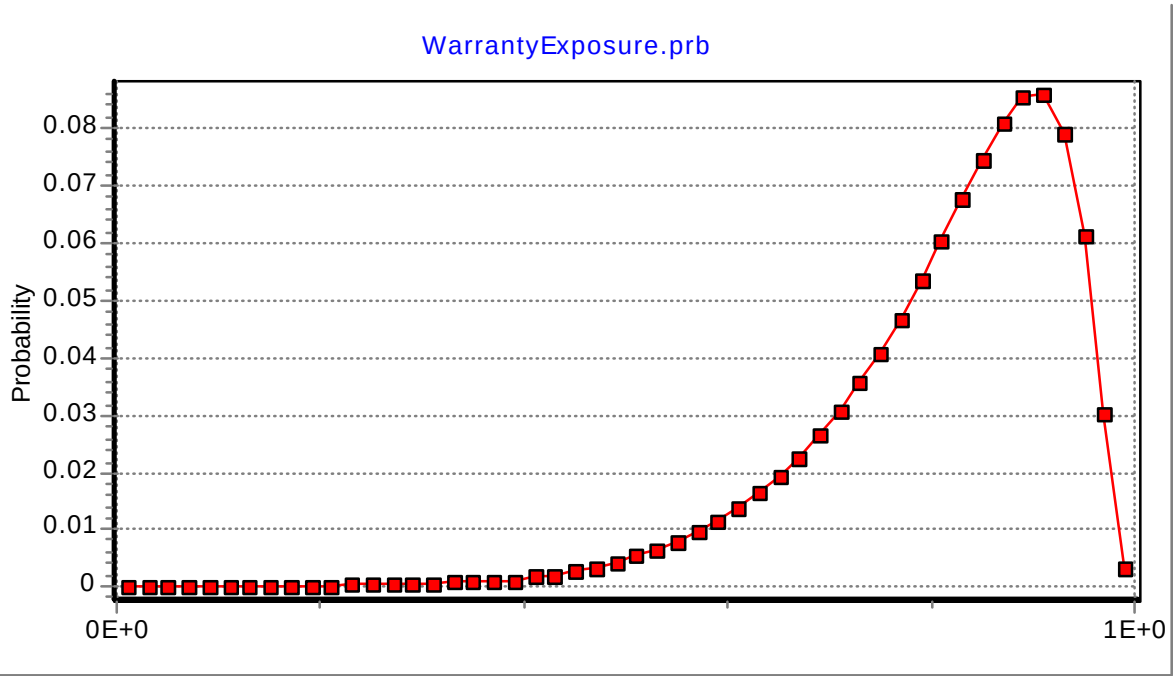


Figure 15.3: Model of warranty exposure with uncertain parameters in Weibull equation. The horizontal axis is the reliability at 900 cycles and the vertical is probability as a histogram.

The result is shown in Figure 15.3. The chart shows that there is a gradual lower tail, and a steep upper tail. The upper tail stops by 1.00, which is consistent with the interpretation of reliability being a maximum of 100%. The mode (peak) is at about 90%, the median is 83%, and the mean is 81%. Compare these to the deterministic reliability of 100 - 18 = 82%. The figure is valuable as it shows the possible spread in reliability. The cumulative curve is also produced by the simulation, though not shown here. From it other percentile metrics can be obtained, eg there is an 18% probability of getting a reliability of 70% or lower.

As the design process progresses and prototypes are built and tested, it becomes possible to tighten the confidence limits on the two Weibull parameters. Consequently the above curve for reliability would narrow. It is also possible to replace the warranty time with a full probability distribution, but this is not discussed further in this chapter.

15.3 **Conclusions**

This chapter has demonstrated a methodology to apply probabilistic computation to reliability estimates. Reliability is shown as a probability distribution rather than as a single point value. The inherent assumption is that Weibull shape and life parameters may themselves be quantified in terms of a probability distribution. This issue is addressed by providing a mechanism for using beliefs, as expressed in confidence intervals. Consequently the method is able to be used at early design stages where reliability data are typically sparse. In this way the reliability viewpoint is able to be represented earlier in the concurrent engineering process. Application of probabilistic reasoning to reliability estimates provides a tool for the management of warranty risk and a means to direct design efforts to solve risky areas.

Future work might explore ways in which uncertainty bands for Weibull parameters could be defined in other ways, eg using likelihood and or Bayesian beliefs.

Chapter 16

Dishwasher hazards

This chapter develops a fault tree approach to anticipating dishwasher hazards and safety at early design. Uncertain failure events are represented by probability distributions, and probabilistic computation is used to determine the probability of the top event occurring.

16.1 **Dishwasher safety and hazards**

Safety is an important issue in the development of dishwashers and other consumer appliances, because of the significant financial claims that can be made against the manufacturer of a faulty product. One of the fundamental principles of product liability is that the manufacturer, being in the position of most directly affecting the product safety, has a duty of care to avoid harming other people. The prudent manufacturer is therefore interested in determining the safety of a product as early as possible in its design.

The term *hazard* is used here to refer to events that cause injury to people or damage to property. The main hazards for consumer appliances are electric shock, fire, and injury as shown in Figure 16.1.

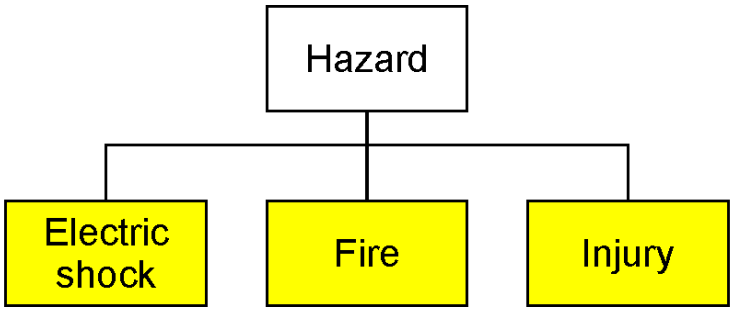


Figure 16.1. The main hazards for consumer appliances are electric shock, fire, and injury, as this fault tree shows.

These hazards are important as their existence in a product increases the risk of the manufacturer being liable to legal action. It should be observed that the term *hazard* has various usage in reliability engineering. Elsewhere it is a measure of the probability of a device failing, whereas here it refers to a dangerous failure mode, perhaps without any explicit probability.

There are several organisations that issue standards regarding safety and hazards. For the USA market the most significant of these is probably Underwriters Laboratory

Inc. (UL)[167], who publish standards by which a product may be checked for safety. The organisation also tests products and materials against hazards to life and property. The main failure modes that concern UL are fire and electric shock. Injury hazards are dealt with in less detail, and perhaps this reflects the generally high level of safety achieved in domestic appliances as regards injury. The UL standards provide a prescriptive approach to safety issues, i.e. they lay down tests that a product must pass, but do not generally discuss methods whereby the product designer can design against the failure. The standards do not take the fault tree approach adopted here, and the principles on which the standards are based are not always apparent in the regulations.

Certifying a product through UL is valuable for the protection it provides in cases of liability. Liability is a serious issue for manufacturers who wish to sell products in the USA, since the legal system can award substantial damages against the manufacturer of an unsafe product. The UL certification itself does not provide any legal immunity, but what it does give is proof that the product has been independently scrutinised and judged to be safe.

Appreciation of hazards in the early designs stages is important, as this is where the greatest flexibility exists to optimise product characteristics. There is minimal literature on methods to incorporate design-for-safety issues at the early design stages, and none whatsoever is known that is specific to dishwashers. This chapter develops a method to address safety issues early in the design. The method is based on fault tree analysis of hazard events. The intent is that the designer would apply the fault tree process in a top-down manner to explore the hazard modes of the product under consideration. Each of the top events in the figure are now developed further.

[167]This is a non-profit organisation dedicated to public safety. As the name suggests, it does represent the interests of insurance companies, but it also includes many other interest groups. The UL standards are developed in consultation with a wide group, including manufacturers, consumers, USA government, and insurance interests.

16.2 **Electric shock**

The hazard here is the risk of electric shock to the user or the serviceman. In order for an electric shock to occur, there are two principal events that both need to occur. These are that (1) there must be access to parts that are potentially dangerous, and (2) the hazard must exist at the time, and these are represented in Figure 16.2. For example, for electric shock there must be direct access to conductive parts, and those parts need to be carrying current at the time. The fault tree for electric shock shows these two events, with the *'and'* gate (both events must occur). All other junctions in these fault trees are to be interpreted as *'or'* gates (any one of the events can occur) if nothing is stated. The following fault trees progressively explore each of these main aspects.

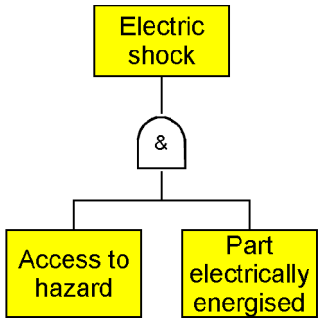


Figure 16.2. Electric shock requires that two events occur, Access to the hazard, and that the Part be electrically energised.

Access to hazard

There are several failure modes that can occur here as shown in Figure 7.3, any one of which is sufficient on this part of the fault tree. The first is access to *internal surfaces*. The assumption used by safety certification organisations such as Underwriters Laboratories (UL) is that a user might push a finger or a knife into the product, and thereby touch something live. To the engineering design this tends at first to seem an unreasonable thing to have to design against. However it needs to be noted that children will sometimes explore household appliances in a potentially dangerous way due to ignorance. Adults also are quite capable of using a product in novel ways. For example an article may have fallen inside the machine and the user attempts to remove it with a finger or a knife. The UL standard (UL 749 Household dishwashers, 1997) tests for accessibility by using probe implements that simulate fingers and a knife, unofficially called "test fingers". The geometry of these implements is precisely defined in the standard. The probes are prodded into the machine from all angles, as far as they will go, both inside the wash cavity and

outside. The probes must not touch internal wiring, nor be able to displace any metal part to touch an uninsulated current-carrying part.

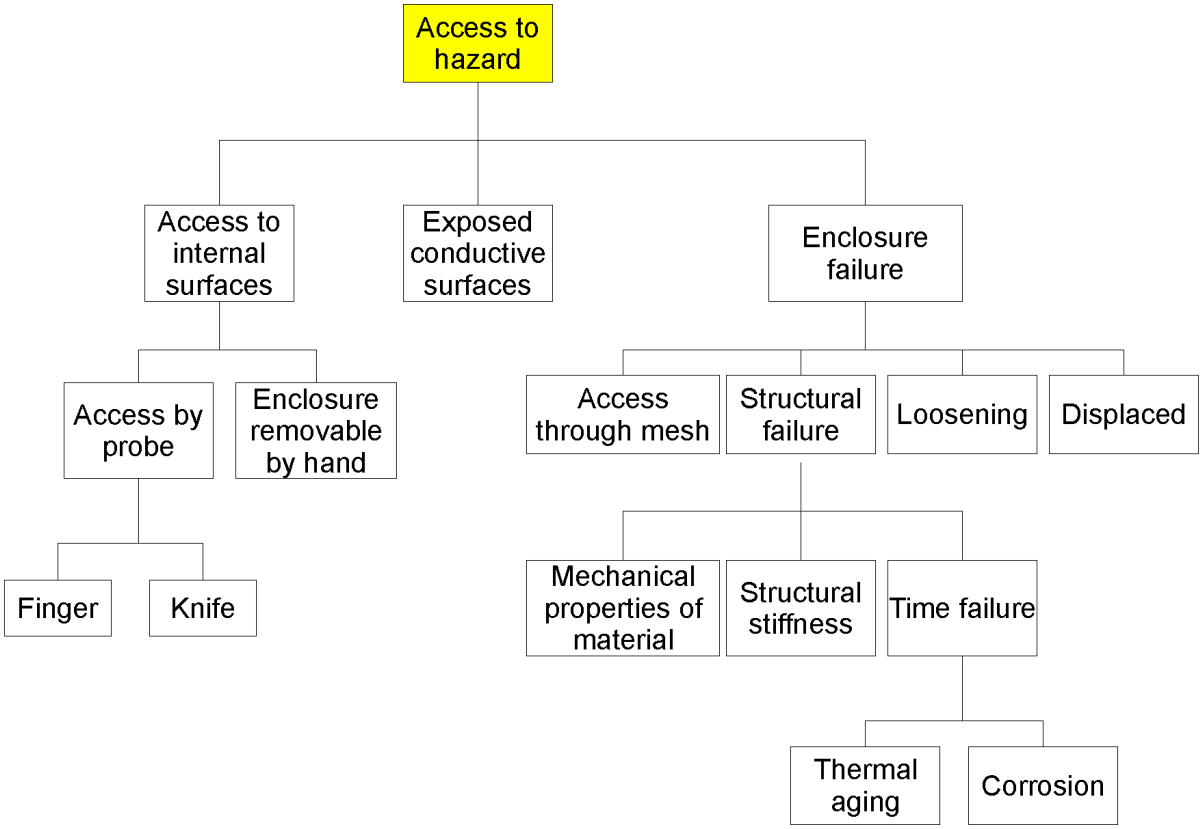


Figure 16.3. Access to the hazard is one half of the electric shock hazard. This fault tree shows various potential causes whereby a person can gain access to an electrical hazard.

Another hazard mode is the user removing an enclosure or panel on the machine. The critical test is whether an enclosure can be removed without the need for tools. If an enclosure can be removed by hand, then it could provide access to parts that cause shock. The UL requirement is that a product be sufficiently well enclosed so that internal current carrying parts are not accessible. If an enclosure can be removed without using a tool, then it must be removed for the accessibility test.

Exposed conductive surfaces are another area of risk from electric shock. These parts may become energised by an internal fault in the product. Most large domestic appliances have metal wrappers, and these are of course exposed conductive surfaces. However it is not usually a problem so long as the wrapper is properly

earthed. More problematic are plastics that have been metal coated, thereby also becoming electrically conductive. These plastic parts are usually difficult to earth, and instead it may be necessary to ensure that they do not receive too much leakage current in the first place.

Enclosure failures are those where access is provided by failure of the protective structures. If the enclosure is in a mesh form, then one of the failure modes is a probe being passed through the holes. Structural failure of the enclosure can also lead to access to current carrying parts, and in this regard the issues are the strength of the enclosure material, thermal aging, and corrosion. Enclosures may also fail by loosening, or by being displaced.

Part electrically energised

The other necessary half of the electric shock fault tree (refer Figure 16.2) is the part being electrically energised, and lower level faults for this are illustrated in Figure 16.4.

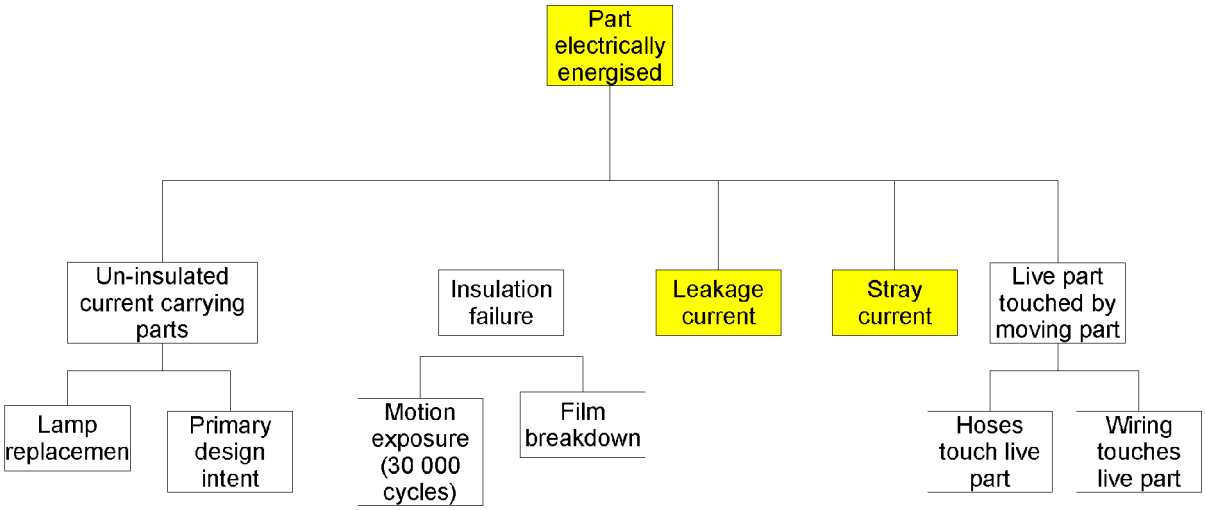


Figure 16.4. Electrically energised parts are necessary for electric shock to occur. This figure shows contributory causes.

The simplest case where the hazard is always presented is *un-insulated parts*, such as terminals, inside the machine. Of course the uninsulated parts are inside the machine by design. However any one of the access events may have occurred (eg a knife entry) so that the user is closer than the designer expected to the un-insulated

part. *Insulated parts*, including internal wiring, can also present the hazard, if the insulation is damaged. The fault conditions where this could happen are with film insulation, and motion exposure. Film insulation is a thin layer that covers wiring such as motor windings. Due to its thinness, there is a greater possibility of the insulation failing. Wire with thicker insulation also has risk of failure, particularly if the wire moves during the normal operation of the machine since the motion can cause fatigue failure of the insulation. The discussion about un-insulated and insulated parts concentrates on whether or not the electrical insulation is physically present. There is another fault mode that needs to be addressed, and that is leakage current. *Leakage current* concerns current that escapes through electrical insulation and appears at other conductive parts. Figure 16.5 shows how Leakage current requires that an electrical leak source exist, and also that the earthing on the part be inadequate. Each of these have lower level events. As regards electrical leak, there are several sources: insulation failure, over voltage of the supply, and one of several device failures. The list of device failures is illustrative and is not exhaustive.

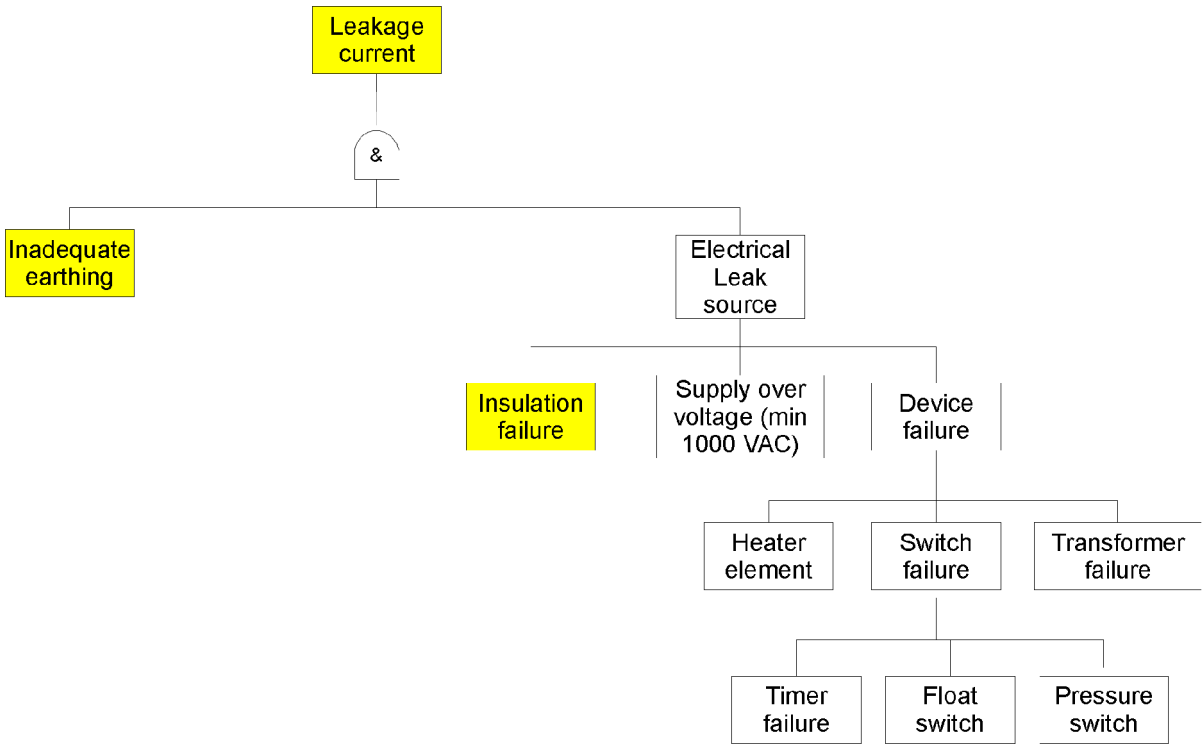


Figure 16.5: Leakage current requires that an electrical leak source exists, and also that the earthing on the part be inadequate. Each of these have lower level events.

For example, the UL requirements for dishwashers are a maximum leakage current of 0.75 mA. The product must also pass a high voltage breakdown test (1000 V, 50 Hz).

Inadequate earthing is necessary for dangerous leakage current to occur on a product. If the earthing is adequate then any leakage current would be conducted harmlessly to earth. Some of the causes of poor earthing are shown in Figure 16.6.

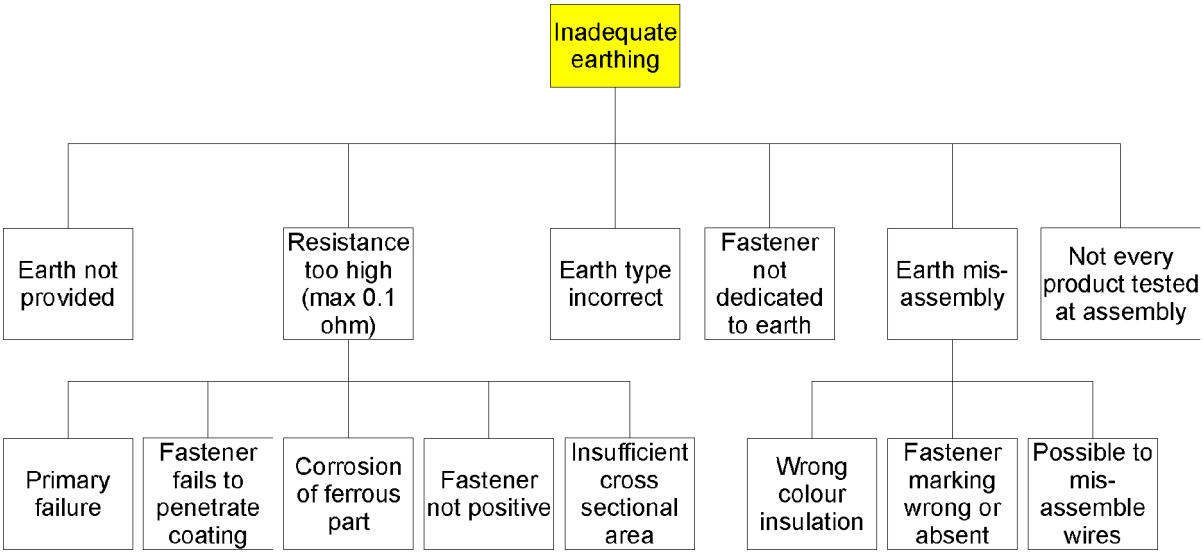


Figure 16.6: Inadequate earthing has a number of causes, any one of which can cause the fault.

Insulation failure (Figure 16.7) is another cause of an electrical leak[168]. Under insulation failure are several other causes, particularly primary failure of electric strength, wetting , and insulation damage. The primary failure is inadequate electric strength of the insulation, which can be attributed to design fault. As far as domestic

[168]In the UL tests, insulating materials are tested for leakage current when subject to several conditions, specifically:

- High humidity.
- Liquid overflowing from detergent dispenser or rinse aid dispenser must not wet any electrical component or compromise electrical insulation.
- Oversudsing (excessive foaming) must not wet any electrical component or compromise electrical insulation.
- Liquids could be spilled onto the product, and must not wet any electrical component or compromise electrical insulation.

appliances are concerned, the next most important faults to guard against are wetting and insulation damage.

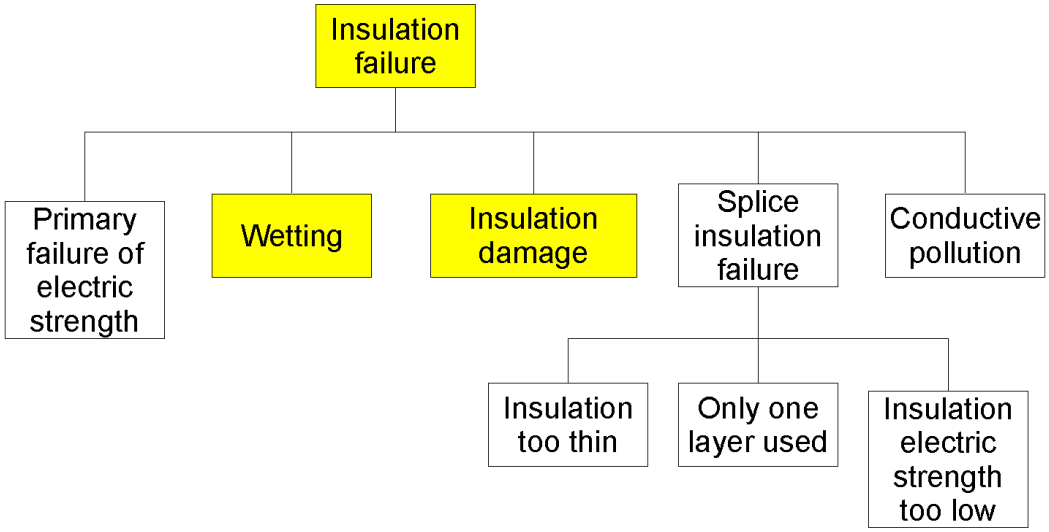


Figure 16.7: Fault tree for Insulation failure

Wetting is a realistic possibility for appliances that are used in the kitchen. Figure 16.8 shows some of the causes.

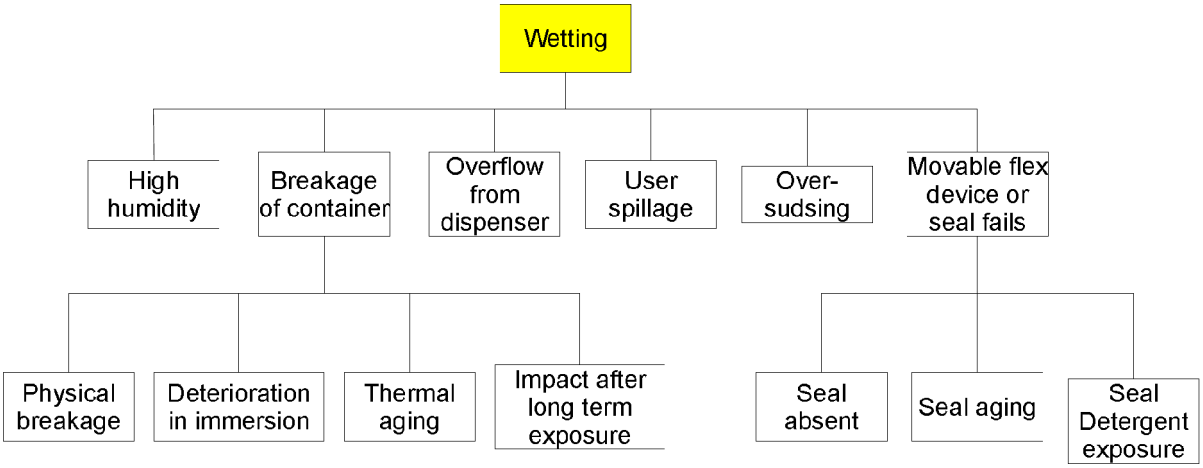


Figure 16.8: Wetting of a domestic appliance is a real possibility that needs to be designed for. Various foreseeable fault conditions are shown here.

The UL tests contain specific provision for wetting tests and liquid spillage. A test volume (200 ml) of water is poured onto the appliance, with the door in open and closed positions. Sealing failure is simulated by requiring that all rubber parts, boots

and seals shall be removed and the appliance operated through one complete wash cycle. Overfill is tested by defeating the timer switch or level sensor (float or pressure operated switch) and the appliance started in a fill cycle. The fill shall be continued for 15 min after the overflow of the wash cavity, unless a second flood sensor terminates the fill earlier. After each of these tests the product shall comply with stated requirements for electric strength (no breakdown for 1000 VAC applied for a minute) and insulation resistance.

Insulation damage concerns the damage to what would otherwise have been acceptable insulation. There are several causes shown in Figure 16.9. Deterioration of the insulation would usually be an obvious fault to guard against in design. Less obvious is the need to design against flexural and mechanical damage to the insulation.

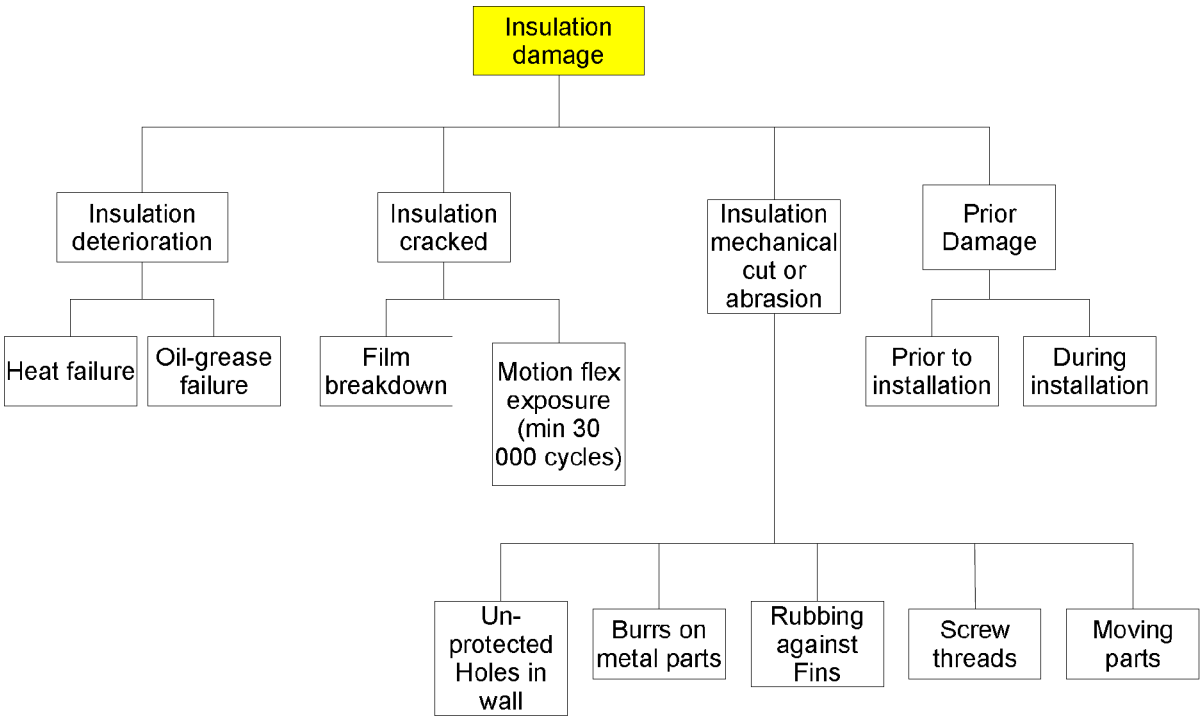


Figure 16.9: Insulation damage involves several possible causes, of which damage due to mechanical cause needs to be an important design consideration.

Motion flex of a wire is considered seriously. For example the UL tests will fail a product if a probe is able to touch any movable or unsecured wire, even if it is insulated. The probes are only allowed to touch insulated wires if the wiring is

secured so that it is not subject to stress or mechanical damage, or if it is double insulated.

Stray current

Moving back to a higher level, another cause of a part failing into a state where it is electrically energised, is the presence of *stray current*. The causes of stray currents are shown in Figure 16.10.

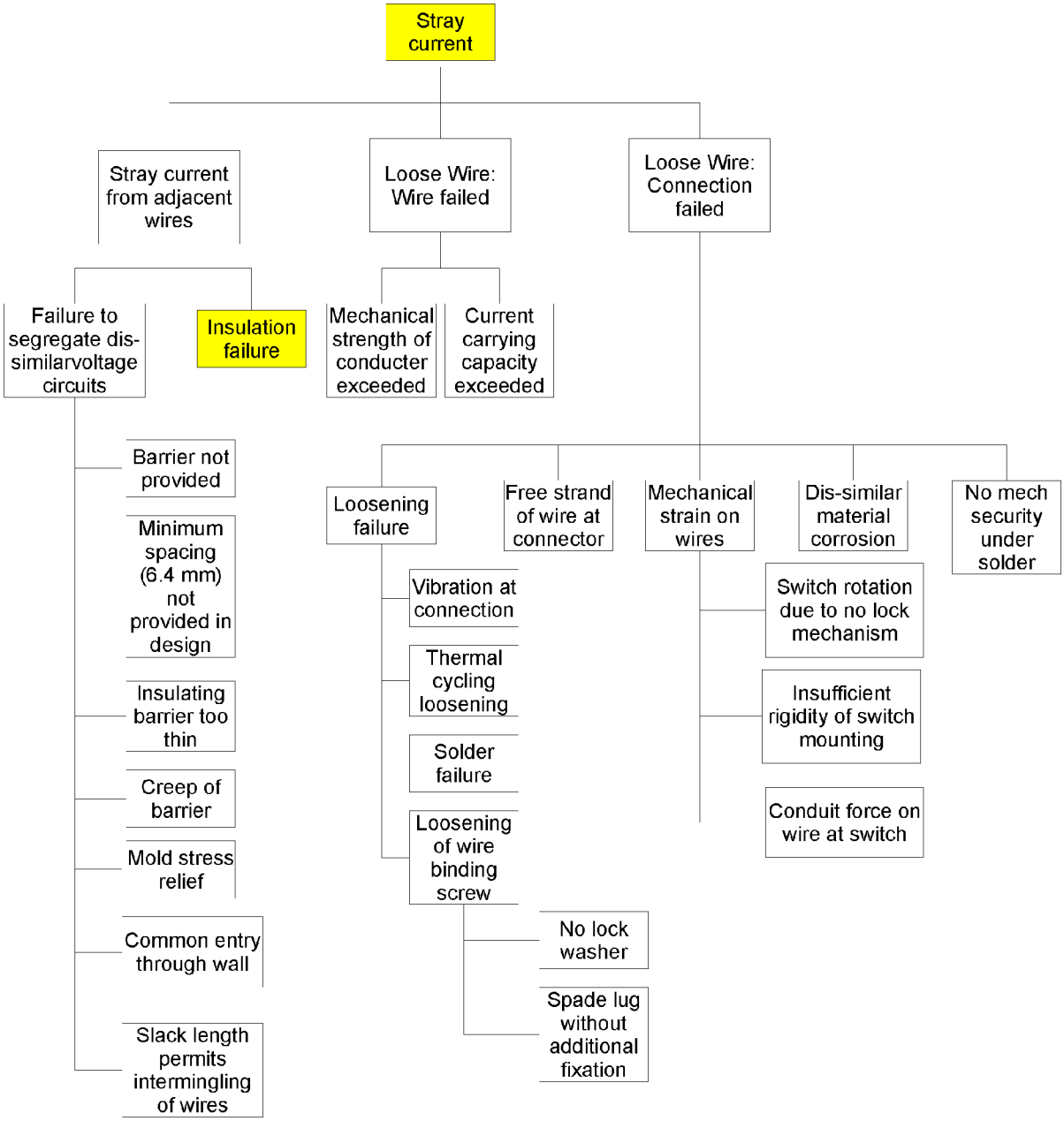


Figure 16.10: Stray current can be caused by one of several fault conditions.

There is some overlap between stray currents and leakage current, in that similar faults can cause either or both of these conditions. Stray currents are particularly critical since they can cause injury and also loss of control of the performance of the product. For example, stray currents are sometimes implicated in aircraft crashes where systems have operated out of normal control. Sometimes designers appear to view product performance in a deterministic fashion, namely that the product will behave exactly as specified, and they may find it difficult to acknowledge the possibility of stray currents. However stray currents are relatively easily ascribed to two main causes: current passing from high- into low voltage circuits (due to insufficient insulation barrier), and loose live wires that have come off their connections.

16.3 **Simulating shock hazard at early design**

The above fault trees provide the context in which fault trees were semi-automatically generated using the Design for System Integrity (DSI) methodology. As described in earlier chapters, the methodology uses multiple viewpoints, and the software embodiment is able to automatically create the basic data in other views. For this application the domain file included *electric shock* data. These were representative data and gave the time [in years] at which the failure event would occur (eg the age at which the device would have deteriorated so as to be electrically energised).

The devices were then connected up with relationships. A simple version of the above fault trees was modelled, as there is insufficient data to attempt to model all the complexities. The model is shown in Figure 16.11.

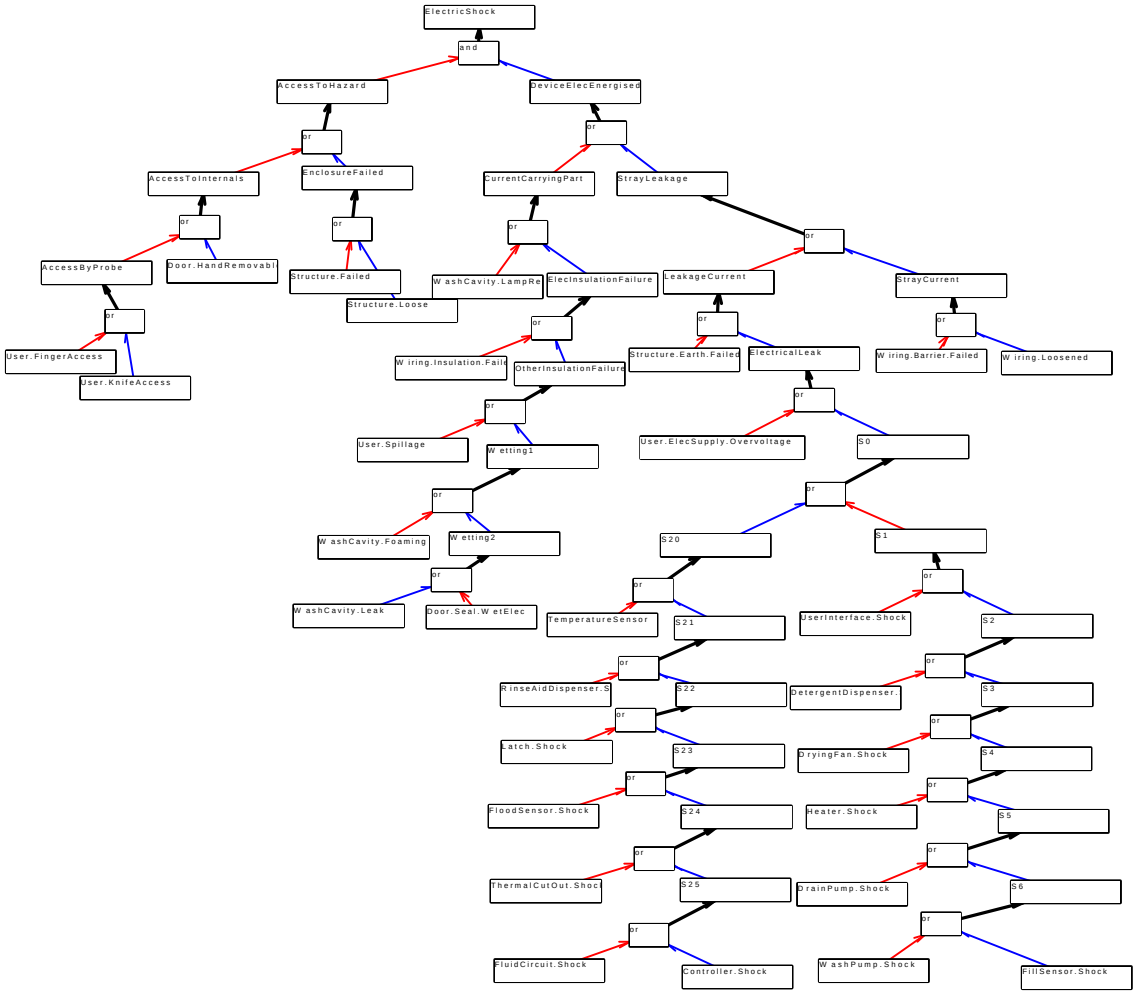


Figure 16.11: Model of electric shock hazard, as expressed in DSI. Default data (purely representative) were provided for each of the prime events.

Conventional relationships for fault trees are ‘and’ and ‘or’, and these were used. In probabilistic terms ‘and’ corresponds to the ‘max’ function, and ‘or’ to ‘min’. Probabilistic computation was then performed using the DSI engine, to determine the probable time before someone received an electric shock. The final result is shown in Figure 16.12.

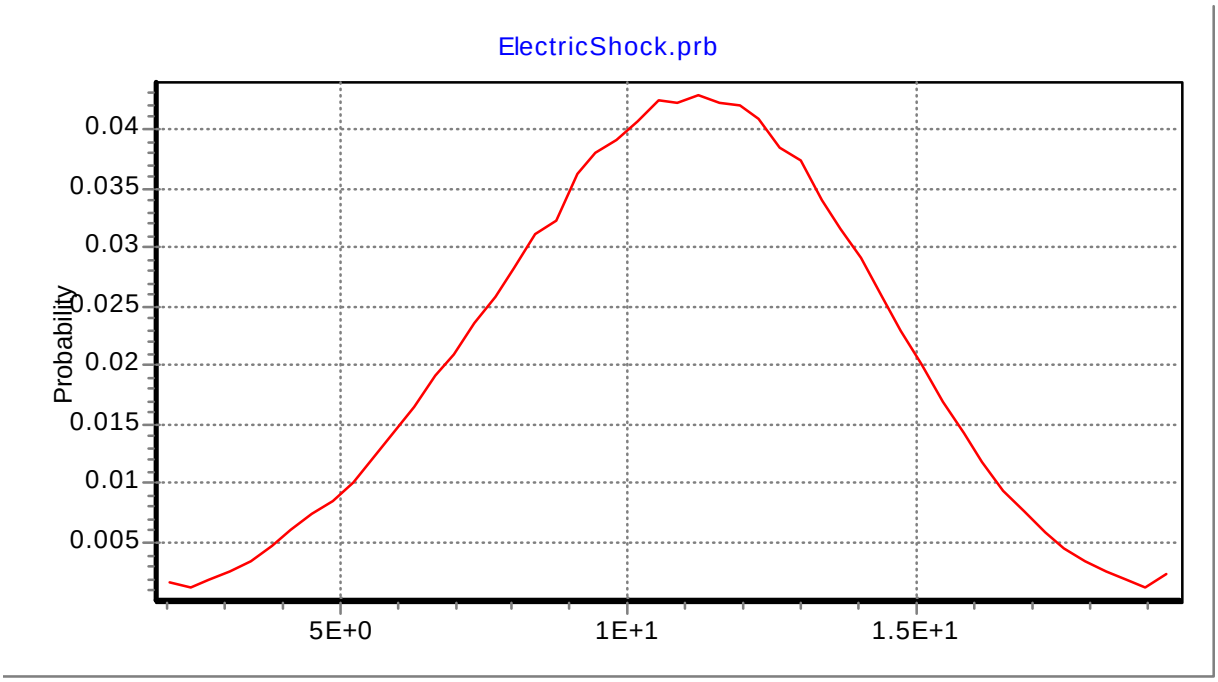


Figure 16.12: Results of probabilistic computation of electric shock hazard, giving time to this event. The horizontal axis is time [years] and the vertical axis is histogram probability. The model predicts a mean time of about 11 years for an electric shock to occur, but it should be remembered that the data used here were only representative.

16.4 **Fire hazard**

The *fire* hazard exists when two events occur: an ignition source exists, and combustible material is present. The fault tree in Figure 16.13 shows the contributory causes. The existence of an ignition source can arise from either a part failing in a thermally aggressive way, or from electric spark. The causes of electric sparks are covered previously under the topics of leakage current and stray current.

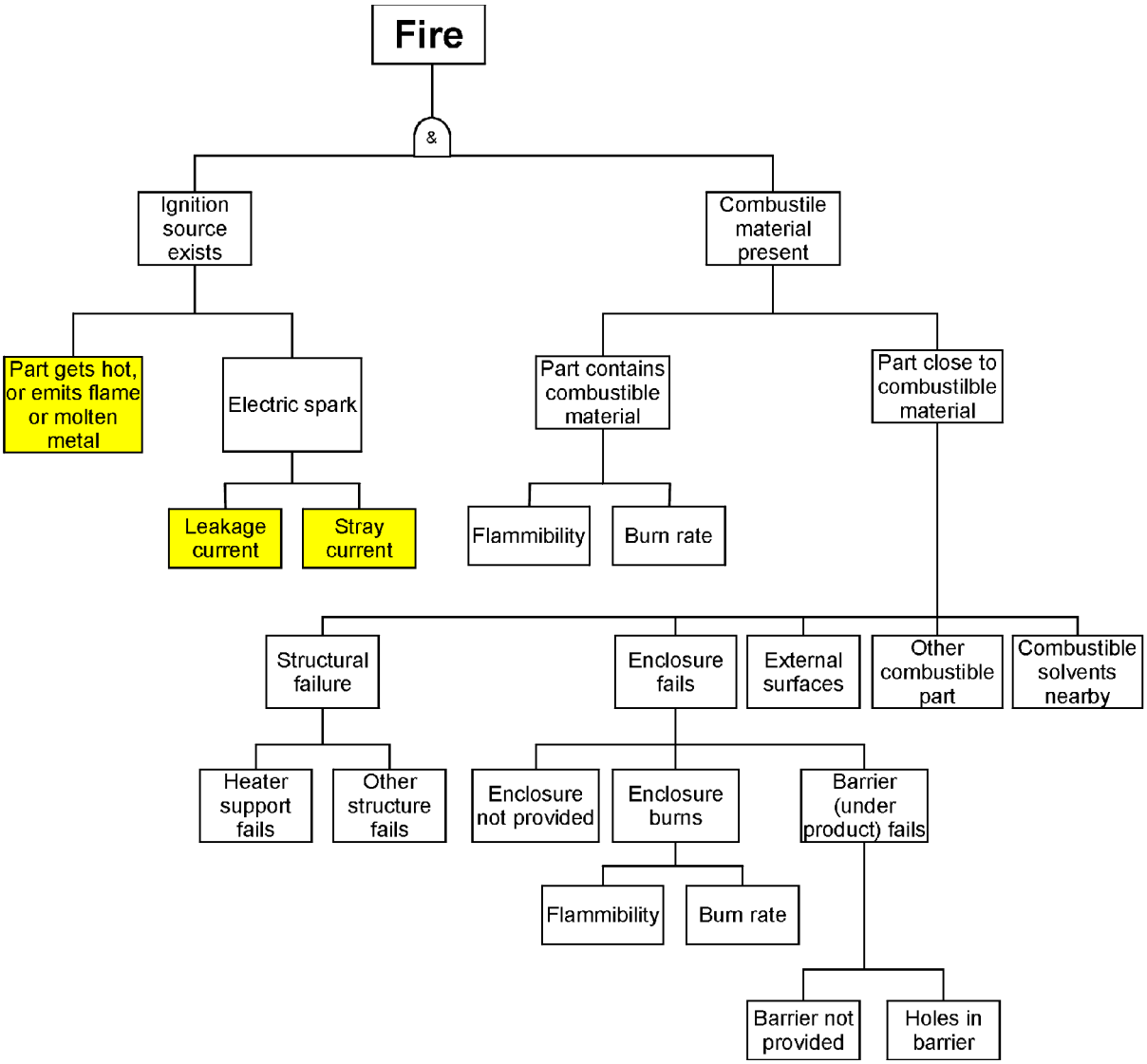


Figure 16.13: Fire fault tree

Moving to combustible material on the other side of the fault tree, the fault events to guard against here are the part itself containing combustible material, or being too close to combustible material. As regards the part itself, the principle tests involve flammibility and burn rate. As regards being close to another combustible material, the main design provision is a suitable enclosure or barrier to contain the thermal event. This enclosure can be made of plastic, providing that it meets requirements for flammibility and burn rate.

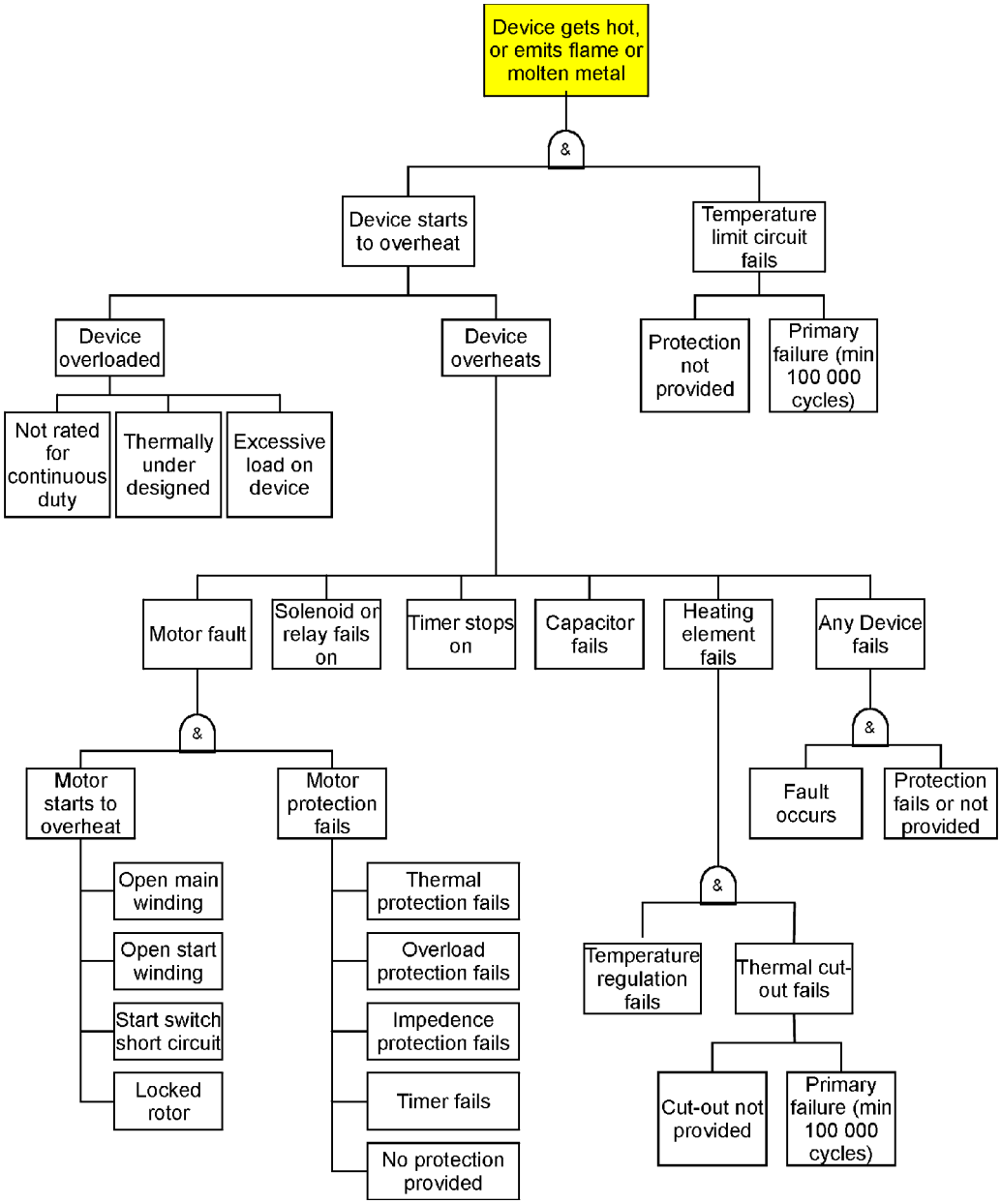


Figure 16.14: Fault tree for causes of a part becoming excessively hot

Tests such as UL will fail a product if a device gets excessively hot, emits flame, or molten metal. If a device fails in such a way, then it needs to be enclosed. There are a number of levels of faults for these cases, as shown in Figure 16.14.

The designer has to consider the possibility that failure elsewhere might cause the device in question to be exposed to full voltage continuously. This effectively means

that devices have to be rated for continuous duty even if they are only used intermittently[169].

16.5 **Injury hazard**

Injury refers to the harm caused to humans, and in this context typically refers to bodily injury or even death. The two necessary components to injury are that a person be in the wrong place, and at the wrong time. These are illustrated in the fault tree in Figure 16.15. Being in a wrong place (a hazardous place) with respect to a machine is related to how the machine is used. On the other side of the fault tree, being at the wrong time (when a hazard actually occurs) requires that both a hazard exists and that the prevention system (if any) fails. These aspects are described in further detail in following fault trees.

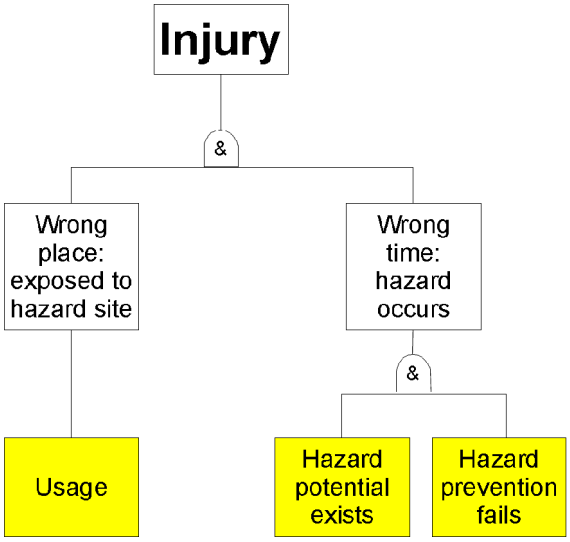


Figure 16.15: Fault tree for injury

Usage

The use to which a product is put is a critical factor in whether or not the user is exposed to hazard. Further to that is the issue of who is legally liable for any injuries that arise. The most straight forward case is when the product is used in the way that the designer intended. This is after all the design intent as regards the performance

---

[169] Some of the UL fire requirements are as follow:

- Maximum permissible temperature rises are given for various devices, such as motors, coils, wires, enclosing cabinet, eg 35°C for insulated wires.
- Burnout of Coils (relay, solenoid): shall be wrapped in cheesecloth and supported on tissue paper, and then subject to power for 7 hr (or until it fails open). The device must not emit flame or molten metal or cause glowing of the cheesecloth or tissue paper.
- Stopped timer: timer and thermostats shall be defeated, and no molten metal to be emitted.
- Motors stalled, solenoid armatures blocked open, heater elements with out water, racks placed close to heater element (as close as in normal use).

of the machine, and the user is obviously entitled to use the machine in this way. If the machine fails while being used as intended, and causes injury, then the manufacturer is liable. However other cases are not necessarily so clear.

Figure 16.16 shows a fault tree for usage. The faults have been broadly separated into two types: unreasonable use, and reasonable use. Within unreasonable use are factors such as intoxication, fooling, and negligence of the user. The legal minimum is to design for reasonable and foreseeable use of the product, so generally the designer of a machine does not have to consider the events in the unreasonable group. Having said this, negligence on the part of the user may be a difficult event for the manufacturer to prove. In addition, what might be negligent behaviour from the manufacturer's perspective might not be so from the user's or from society's.

Moving attention now to the faults in the reasonable group. Essentially everything that is reasonable and foreseeable usage of the product needs to be accommodated. It will be noted in the figure that there are many usage conditions other than the plain intended function. It is precisely when considering hazards related to reasonable usage that the interpretations of engineers and lawyers differ most. Engineers and manufacturers may believe that they should only be responsible for hazards that occur under the intended usage of the product, whereas the law holds them responsible for a broader collection of hazards. The general guideline is that if the use was reasonable, then the law of product liability will tend to find against the manufacturer.The hazards in the *reasonable* group should at least be given scrutiny by the designers.

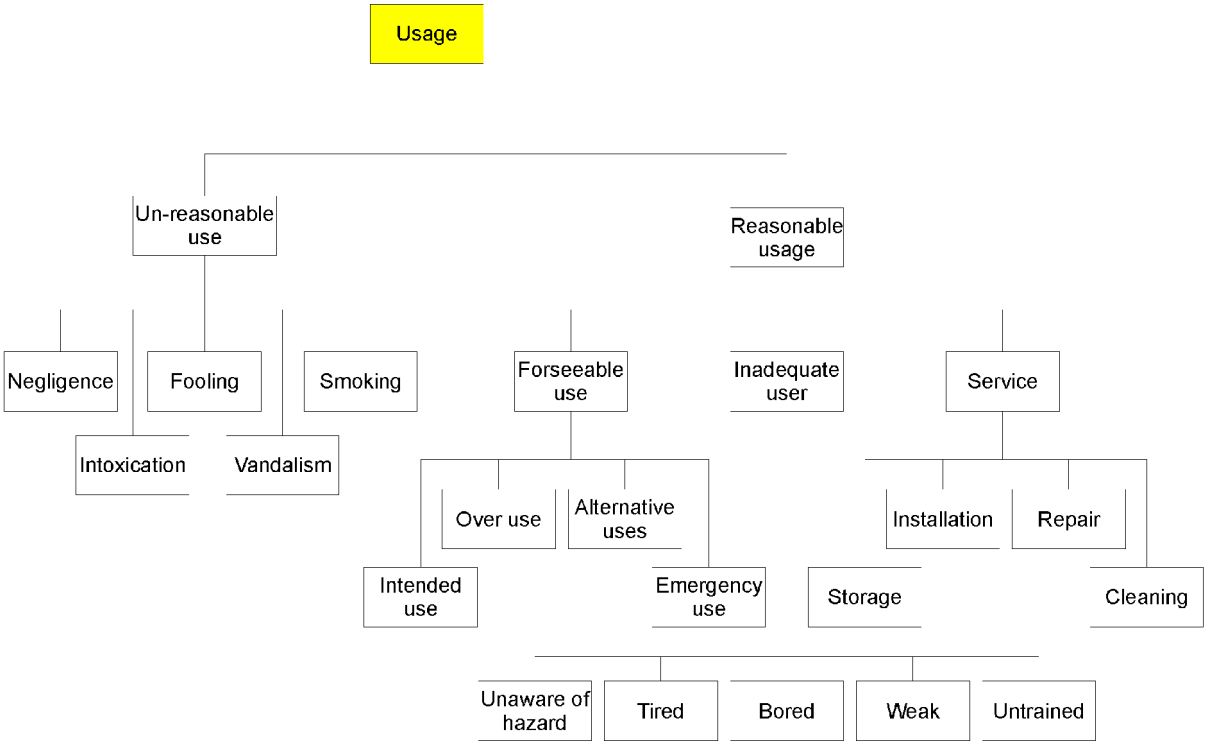


Figure 16.16: Fault tree for usage

Among these are the need to accommodate hazards that may arise due to over-use, alternative use, and emergency use. Of these the alternative usage category is the one most problematic to product designers. Other groups of hazards that need to be considered are users who are inadequate in terms of some criterion (eg physical strength) to use the machine in the manner that the designer intended. The last group of hazards are those related to servicing and installing the machine.

Attention is now directed to the other branch of the fault tree, which is being at the wrong time, when a hazard occurs. The first aspect of this is that a hazard potential exists, and this is illustrated in Figure 16.17.

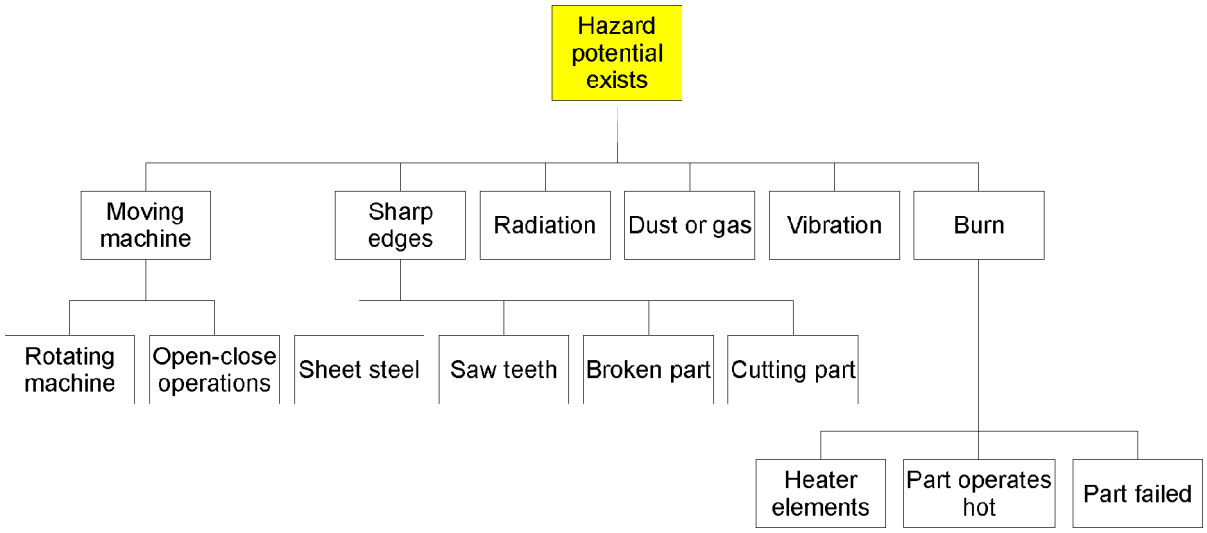


Figure 16.17: Fault tree for the existence of potential hazards.

The main hazards in domestic appliances are moving parts, sharp edges, and burns. The hazard of electric shock is deemed to be so significant that it is treated as a hazard on its own. Most appliances have motors and sharp edges, but generally they are well enclosed inside the product. Heating elements are more difficult to enclose in many domestic appliances.

The UL standard address mechanical hazards in a number of ways:

- Stability of portable devices: these shall not overturn during intended use, tested with a 23 kg weight at the outer edge of the open door.
- Sharp edges: no risk of injury to persons during use or maintenance.
- Automatically restarting motor not to result in injury, may need Interlock to prevent this.
- Moving parts: shall be guarded to reduce risk of injury. Degree of protection is left to judgement of the designer.
- Opening door during operation needs an interlock as this action exposes moving parts, water splashes out, escape of steam is possible, and exposes heating element. UL defines a reliability test for interlocks.

16.6 **Hazard prevention**

Any one of the potential hazards does not become a hazard event until the hazard prevention system fails. Hazard prevention is shown in Figure 16.18, as comprising prevention systems, safety systems, and warnings.

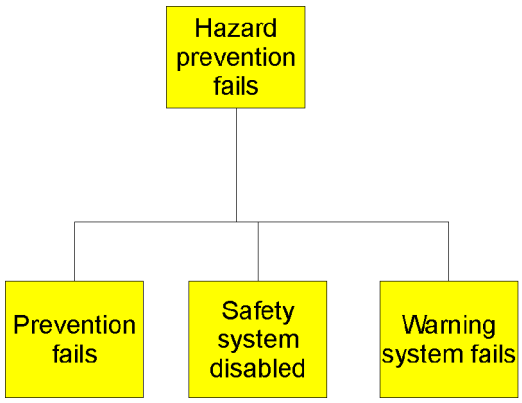


Figure 16.18: Fault tree for hazard prevention failure.

Failure of the prevention system is shown in Figure 16.19. Several failure modes are shown here, and they all need to be considered in the design. The one that merits special discussion is when no prevention system was provided in first place. This can be because it was unanticipated, which in a legal sense can sometimes be interpreted as synonymous with negligence. Omitting to provide a hazard prevention system can be acceptable if it would drastically reduce the product usefulness.

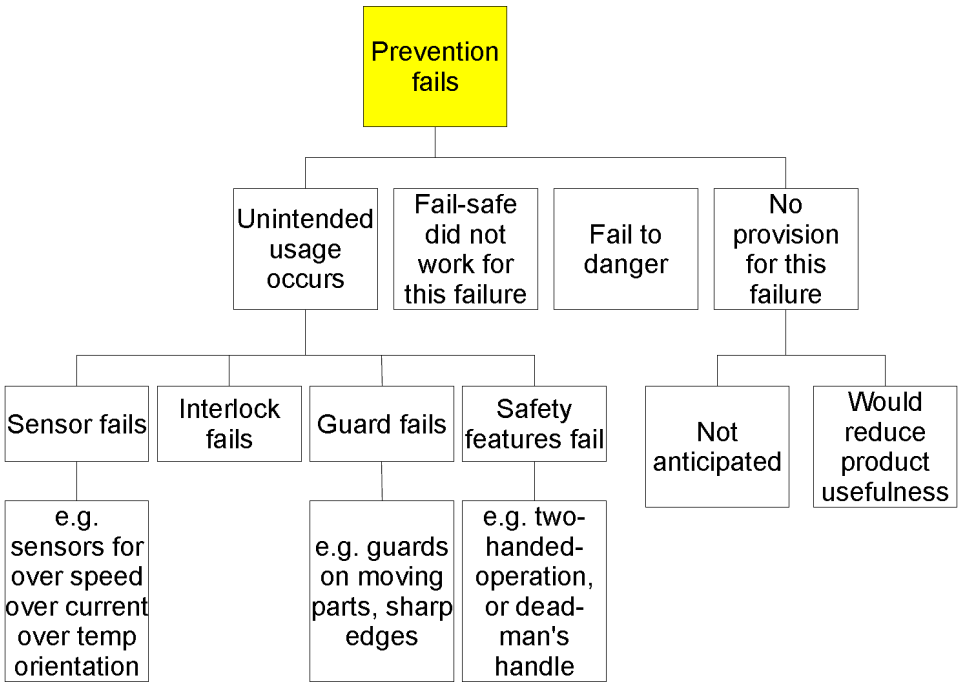


Figure 16.19: Fault tree for failure of hazard prevention systems

If potential injury is serious, then safety features are necessary, even if the probability of the accident is low. It is often possible to add layer on layer of safety features, for example guards, plus interlocks that prevent product operation when the guards are removed, but these add expense. The designer and manufacturer make the decision as the to balance between safety and expense. However ultimately it is the courts that decide on the acceptability or otherwise of the design if called to do so.

Hazards may also come about if the safety system is disabled. The fault conditions are shown in Figure 16.20. Generally if a system is disabled for repair, and a hazard event occurs, then the liability tends to rest on the serviceman rather than the designer. This is not to say that the designer of a

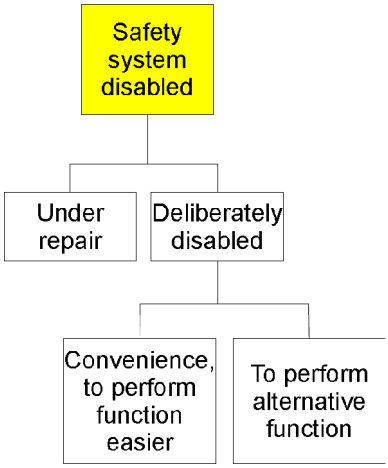


Figure 17:20: Fault tree for safety system disabled.

product can neglect the safety of those servicing it, but can design with a more technically aware person in mind.

Safety systems can sometimes be disabled by the users, either to perform the intended function easier, or to perform different function altogether. The product designer needs to consider these factors, as proportioning liability in such cases can be disputable.

Warning systems are the last group of hazard preventing devices discussed, and in some ways one of the most important though least understood. A fault tree for warnings is shown in Figure 16.21.

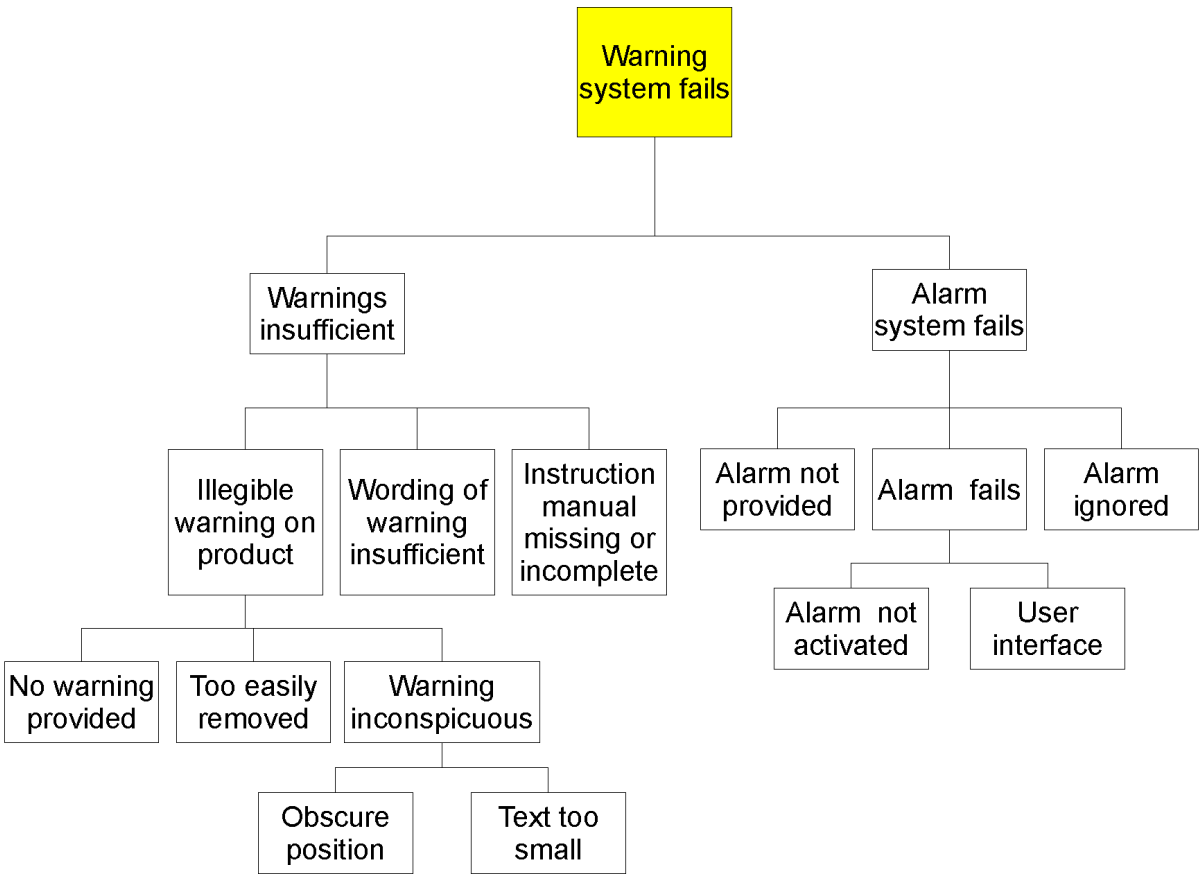


Figure 16.21: Fault tree for warning system failure.

Warning systems are here taken to include written warnings and also alarms/lights. The discussion will concentrate on the written warnings as this is the more contentious area.

Warnings must be put on products that are hazardous to use, or which are dangerous if misused. It is not necessary to warn if the dangers are obvious, or already known to the user. The function of a written warning is to inform the user of the potential risks associated with using the product, so that the user can make up his own mind as to whether or not he choses to use the product under those risks. A manufacturer who inadequately informs the user of risks, effectively forces the user to take unknown risks. The manufacturer has no control over whether the user actually reads the warnings. However a user who of his own decision does not read the warnings, effectively loses the protection of the law. The onus is on the manufacturer to provide warnings that are adequate. Adequacy is defined in terms of a warning that is in a conspicuous position, with text that is large enough, and which is permanently fixed to the product. A warning should be fixed so that it will not be removed by rubbing, or normal operation of the machine (eg detergent must not remove a label on a dishwasher). The wording of the warning also needs to be suitable. Instructions also need to be provided with the product, for the same reasons. The UL standard includes a substantial section on markings, labels, and instruction manuals. It provides test methods to determine how well an adhesive label stays on. It also prescribes phrases which must be present on warning labels and inside the instruction manual.

16.7 **Product Liability**

A discussion on hazards would hardly be complete without considering *liability*, the consequence of a hazard event. This section of the discussion is motivated by an excellent text by Enghagen (1992). Salient points from that reference and others, together with personal observations, have here been converted to fault tree notion and interfaced to the hazard fault trees to provide a holistic approach for designers.

16.7.1 **Cause-consequence model for product liability**

Product liability occurs when a product fails, and causes harm. Both events have to take place for liability to occur. If a hazard or defect does not cause harm, then there is generally no liability. These relationships are shown in Figure 16.22.

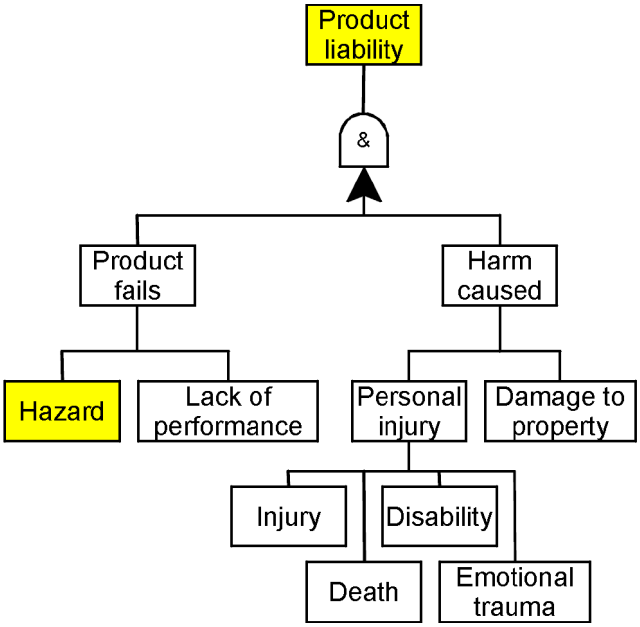


Figure 16.22: Consequence diagram for product liability.

One of the circumstances under which a product may fail is when it becomes hazardous. The principle hazard modes are electric shock, fire, and injury, as discussed previously. Another cause of liability is when a product fails to adequately perform its primary function. This kind of failure opens the manufacturer to claims of

misrepresentation. The types of harm that attract damages are personal injury and damage to property.

Once the ingredients of product liability exist, then a set of consequences are set in motion. These are shown in the *consequence diagram* of Figure 16.23. This has the same structure as a fault tree, but it should be read from the bottom upwards. The consequence diagram uses *and* junctions in a sense consistent with fault tree notation, which is to show that all the lower events must take place for a higher event to occur. Other junctions are *or* gates, which mean that any one of the underlying events can cause the event.

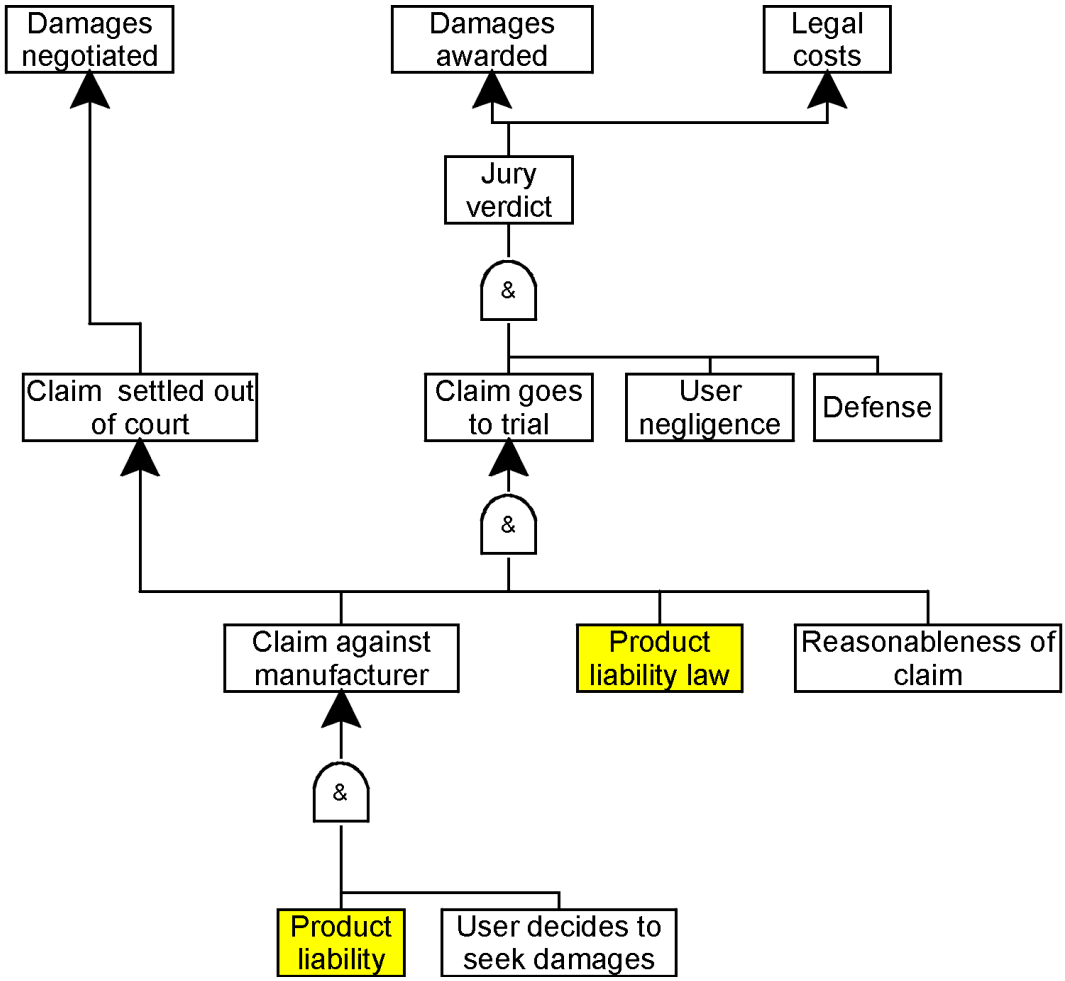


Figure 16.23: Consequence diagram for the process of product liability.

The first event of significance is whether the user decides to seek damages or not. A number of factors can influence this decision. Such factors include: extent of damage, perceptions as to how the claim was first handled by the manufacturer, manufacturer's financial assets, user's expectations, price paid for product, and culture of the particular society towards product liability[170]. Generally product liability falls on the manufacturer, or the local representative (importer or supplier) of a product made in another country. Individual design engineers are also liable, though they are not usually named in the suit. This is because in most cases the assets of the individual are insignificant compared to the company. A situation where this is not the case is when the designer works for a small company or in private practice, and here the individual may be more at risk. In the discussion that follows the term manufacturer will be used, though it should be understood that this could include these other parties.

At the time a claim is made, the manufacturer can reject the claim out of hand, or seek to negotiate the settlement. If the claim can be settled out of court, then that concludes the matter. If this does not occur, then the product user can elect whether or not to bring a lawsuit against the manufacturer.[171]

---

[170]Enghagen (1992) writing about USA product liability, states that "product liability lawsuits generate more $1-million plus verdicts than any other type of personal lawsuits except medical malpractice cases" (p1). Regrettably it seems that these substantial verdicts make product liability claims attractive cases. Also the cost of mounting a product liability claim is not necessarily an obstacle to the USA user, since the lawyer commonly only charges fees if the case is successful.

[171]The manufacturer's decision as to whether or not to settle out of court depends primarily on the value of damages being claimed, and the legal grounds for doing so. If the damages are excessive in the eyes of the manufacturer, then there may be merit in disputing them in court. The manufacturer also has to assess the legal grounds of the claim, and consider the likelihood of successfully defending the action. Discussion shall return to the topic of the legal grounds for a product liability claim later. If negotiations between user and manufacturer cannot be brought to a satisfactory conclusion, then the user may take the manufacturer to court. Legal proceedures vary between countries. In countries such as the USA, a jury is used to determine the reasonableness of the claims and counter claims, and to decide between the parties. The jury is also responsible for deciding on the damages awarded. The USA legal proceedures, as well as methods to prevent and defend litigation, and product liability law in general are described by Enghagen (1992).

16.7.2 **Legal principles of product liability**

Product liability cases are civil rather than criminal lawsuits[172]. Consequently damages rather than punishment are dispensed. Also, the burden of proof in criminal lawsuits is proof beyond reasonable doubt, whereas in civil lawsuits it is proof by a preponderance of the evidence. It is easier to meet the burden of proof in civil cases.[173] The defect could be a design defect, manufacturing defect, or misrepresentation. A user can claim more than one of these defects. Figure 16.24 shows a fault tree of the legal issues. It is worth observing that products do not have to be perfectly safe, but they must have a reasonable balance between their usefulness, the risks associated with usage, and the cost and practicality of making them safer.

[172]Civil lawsuits are disagreements with other people, whereas criminal cases are disagreements with the state (which represents society). Criminal law is a means to control society, and it provides for fines and imprisonment to punish and deter deviant behaviour. Civil lawsuits are different in that they cannot punish by imprisonment or death: they can only recover the monetary value of the actual losses suffered by the injured party.

[173]Generally the product user (plaintiff) will need to prove all of the following:

(a) The product was defective, and
(b) The defect existed when the product left the manufacturer (the defendant), and
(c) The manufacturer knew or should have known of the defect, and
(d) The defect caused harm to person or property, and
(e) The manufacturer could have done something to reduce the risk.

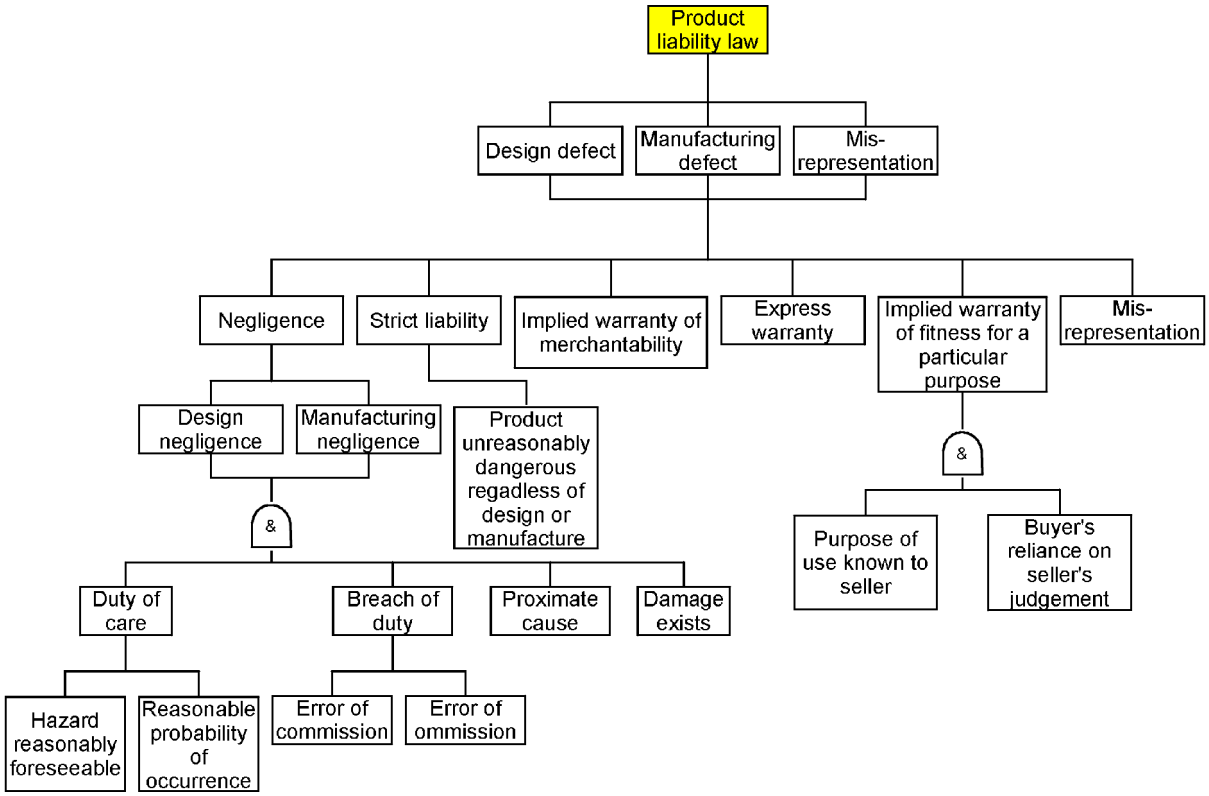


Figure 16.24: Legal principles of product liability represented as a fault tree.

Design defect

A design defect means that the product is intrinsically unsafe due to defects in the design. It has been manufactured correctly, but still fails to provide adequate safety because of the design. A design defect can exist even if the designers followed existing standards, since a user may challenge that these standards are not enough. Contrary to what engineers may expect, the level of safety used in liability cases is not based on standards, but on the principle that the product should be safe for any reasonable foreseeable use. Suits regarding design defects are made in terms of Negligence, Strict liability, and Implied warranty of merchantability.

Manufacturing defect

These are defects in manufacturing, such as the design not being followed properly. A claim of manufacturing defect applies to the product as it was received by the user. It does not necessarily imply that every product is defective, but at least the one received by the user, and on those grounds the user seeks damages. A manufacturer can also be liable if he fails to discover manufacturing defects by reasonable inspection. The manufacturer has a duty of care to exercise reasonable care in producing the product and getting it to the user in a way that it can be used safely. A lack of adequate warnings on the product is also a manufacturing defect, since the warnings are necessary for the safe use of the product. A manufacturer does not have to make a product that is perfect or which lasts indefinitely, but does have a duty of care for safety.

Misrepresentation

This type of product liability applies when the product performance is less than that claimed or stated in the sales literature.

Underlying the claim of defectiveness there needs to be at least one principle of law. The six principles for product liability are negligence, strict liability, implied warranty of merchantability, express warranty, implied warranty of fitness for a particular purpose, and misrepresentation (Enghagen, 1992).

Negligence

This refers to unreasonable behaviour in the circumstances. Negligence can be either commission (doing that which should not be done) or omission (failing to do that which should be done). A negligence claim needs all the following to be proved by the user:

(a) A duty of care exists on the designer or manufacturer, which is duty to act reasonably and prudently and take design precautions against foreseeable hazards in the product. There is no duty if the hazard cannot reasonably be anticipated or its probability is remote.

(b) The breach of that duty by commission or omission.

(c) That the hazard (rather than something else) be the cause of the user's damage, for which the legal term is "proximate cause".

(d) The existence of damage, which can be physical injury, emotional distress, property damage, lost wages etc.

Strict liability

Strict liability applies to a product that is unreasonably dangerous, whether or not there was fault in design or manufacture. It is also called absolute liability. Strict liability is a judgement by society that the product is a menace to society, an unreasonably dangerous product, even though all reasonable care might have been taken in the design and production. The aspect of strict liability that engineers find difficult to accept is that strict liability can apply to a product (eg asbestos) even if the designers were not able to reasonably foresee the hazards. Someone has to be responsible for picking up the cost of the harm done by the product, and as the manufacturer was responsible for bringing it to market, so strict liability is society's decision that the manufacturer should be responsible for the cost of all damages. This may not be entirely fair, but it would be even more unfair to expect the individual user (with much smaller financial resources) to carry the consequences.

Implied warranty of merchantability

Simply by offering the product on the market, the manufacturer implies that it can perform a certain function. This warranty applies regardless of whether or not the manufacturer states it.

Express warranty

If statements are made by the manufacturer or seller about the product, then those become an express warranty. This applies even if the manufacturer does not formally make warranties. For example, statements in sales literature become an express warranty, even if not labelled as such. Sales talk may be an exception, since a reasonable buyer will not necessarily take all sales claims as the pure truth. The user is entitled to a product that meets the promises made by the manufacturer. If the

product that the user receives does not perform as well as represented by the manufacturer, then product liability occurs, even if there is no other defect in the product.

Implied warranty of fitness for a particular purpose

This is a warranty that is implied when the buyer relies on the judgement of the seller, and the seller knows what the buyer needs. If the user relies on his own skill or gets his own expert opinion, then this tends to invalidate the implied warranty of fitness for a particular purpose.

Misrepresentation

Misrepresentation is the manufacturer's assertion about product performance that is not borne out by the actual product. It includes lying or making misleading statements about the product. For product liability to apply, the misrepresentation must be done knowingly, with the intent to deceive, and the user must have relied on the misrepresentation and done so reasonably. As with all product liability, the user must also have suffered actual loss.

16.7.3 Strategies against product liability

Product liability might be better prevented than defended. Manufacturers find it easier to solve product problems with technical solutions at design time rather than take a case to court to ask for society's judgement afterwards. Moreover, damages awarded by courts may be substantial.Two strategies to achieve a safe product are reliability engineering, and organisational proceedures.

Reliability

Reliability refers to the consistency of product performance over time. Most products provide adequate function when they are new, since otherwise they would not be on the market at all. Reliability measures the probability of that function still existing as the product ages. In addition, reliability addresses safety and hazard issues. It

provides methods to analyse how a product might fail, so that unsafe features may be designed out. The detailed mechanisms and methods of reliability analysis are a large topic that will not be discussed in detail at this time, other than to state the need to deliberately and relatively formally address reliability issues during design and manufacture.

Organisational proceedures

Enghagen (1992) discusses strategies for protecting against product liability, and describes the following:

- Inspection proceedures, warnings, disclaimers, instructions. In every case these functions have to be appropriate in a legal sense,
- Tests: whether the intended function is provided, and how reliably,
- Design: foresee hazards, foresee possible uses to which the product will be put, provide safe guards to minimise risk to user, meet standards for safety, materials adequately specified, accidents with similar products taken into account, design review,
- Manufacturing: product made according to drawing, quality control, packaging, labels, hazards originating with bought-in parts,
- Inspection: whether product is reasonably safe,
- Instructions: comprehendible,
- Marketing: product literature consistent with product performance,
- Organisation: quality proceedures in place, quality manuals, committee to co-ordinate product liability loss assurance, compare product reliability against competitors, instruction manuals, risk and reliability analysis,
- Organisation proceedures for: customer complaints, claims, and marketing.

Two ways of avoiding liability are the use of disclaimers and warnings.

Disclaimers

A seller is under no legal obligation to give out any express warranties about the product. Warranties include statements of performance, sales literature, marketing

and advertising, labels on the product, and reliability guarantees. Any such warranties must be consistent with the product as received by the user, otherwise misrepresentation has occurred[174]. Disclaimers are statements used by a manufacturer to negate any implied warrantees about the product. By not promising anything, the manufacturer absolves himself of liability[175]. Disclaimers have to be conspicuous to be valid.

Warnings

The manufacturer has a duty to warn the user of the hazards of the product, so that the user can behave appropriately. The topic of warnings was discussed above and will not be repeated here.

16.8 **Conclusions**

This chapter has presented a fault tree approach to modelling hazards and liability in domestic dishwashers. It has been shown that it is possible to model at least one of the hazards, namely that of electric shock, using the DSI methodology. Other hazards could potentially be modelled likewise. Fault trees provide a simple graphical representation of issues, and can be used in early design stages even if there is no probabilistic computation behind them.

---

[174]If the manufacturer later discovers the product performance is less than that warrantied, then the marketing and other information must be corrected. The manufacturer also has a duty of care to the users of existing products, if an unforeseen serious hazard develops. It may be necessary in these cases to either warn the users, or recall the product. If a recall is not done when necessary, then product liability occurs. The decision whether or not to recall products is based on the risk of product liability versus the direct recall cost.

[175]Express warrantees, which are promises made by the manufacturer, cannot be disclaimed. The implied warranty of merchantability, which is created by offering the product for sale, is the easiest to disclaim with statements like "as is". The implied warranty of fitness for a particular purpose, which is created by the buyer relying on the judgement of the seller, is not as easily disclaimed.

Chapter 17

Discussion

The position of the Design for System Integrity methodology in the design process is discussed and it is concluded that it is an assessment tool. Its analysis capabilities are clarified as being for those problems for which a solution may be expressed explicitly in terms of the output variable. The information content of the results is discussed and it is shown that the methodology produces results with high information content and therefore adds quality information to the decision and management process. Suggestions for future research are provided.

17.1 **Identifying the need for assessment mechanisms**

The preceding chapters described the development of the Design for System Integrity (DSI) methodology and demonstrated the modelling of both quantitative and qualitative[176] uncertainty, and support of multiple viewpoints. Several case studies were given for the dishwasher domain. In moving towards closure this chapter returns to discuss the methodologies for assisting the design process. In particular, where does the DSI methodology fit into the design process? What does it do, and what does it not do?

Chapter 1 presented a model of the development process inside an organisation. The model used IDEF0 notation and successively detailed the inner workings of the design processes. This framework[177] was then used in Chapter 2 to position the available design tools. The following groups of mechanisms were identified: inventive mechanisms, assessment mechanisms, decision mechanisms, implementation mechanisms, and recording mechanisms. Additionally there were two groups of tools used to generate constraints, namely inventive constraints and assessment constraints.

The strategy in this thesis was away from inventive mechanisms and instead towards mechanisms that assist rather than supplant the human designer. In particular the work investigated mechanisms for assessing candidate design solutions. Existing assessment mechanisms were identified as including systems engineering approaches, functional modelling, sensitivity analysis, decision analysis, fuzzy theory, Monte Carlo simulation, and qualitative simulation.

The literature identifies the need for more sophisticated assessment mechanisms that would analyse designs and evaluate solutions (Candy et al, 1996) at all stages

[176]The interested reader is referred to Chapter 6 for a description of how the terms quantitative and qualitative are used in this work.

[177]Refer to Figure 1.11 or Figure 2.19.

but specifically including early conceptual stages (Finger and Dixon, 1989b). The design process is difficult for assessment mechanisms because of the uncertainty and incompleteness of knowledge (Ullman and D'Ambrosio, 1995). There is also the need to assess life cycle information (Rabins et al, 1986; Hague et al, 1996), which is particularly challenging since it involves assessing, and perhaps even anticipating, multiple viewpoints.

17.2 **Evaluating the DSI methodology**

What does the DSI methodology do?

DSI is an assessment mechanism. It permits the functional relationships of a candidate design solution to be assessed by simulating qualitative and quantitative performance from multiple viewpoints.

What is the conceptual contribution made by the methodology?

The DSI methodology helps narrow the gap between the needs of the designer and available assessment tools. It does this by providing a generic modelling tool (i.e. not tied to one specific domain). This tool accommodates the various forms of uncertainty, both process variability and uncertainty of analysis. Process variability refers to a random variable, and the methodology accommodates such variability in both quantitative and qualitative parameters.[178] Uncertainty of analysis refers to incompleteness of knowledge, and the methodology accommodates both mathematically explicit (quantitative) relationships and subjective (qualitative) relationships. The mathematical relationships are handled with one of three mechanisms: Monte Carlo analysis, Fuzzy theory (cut sets), or a discrete integration of joint density specially developed for DSI. The subjective relationships are processed using decision tables. With these internal tools the methodology is able to process both quantitative and qualitative variables even in one model. The methodology supports modelling from multiple viewpoints.

[178] Quantitative parameters are measurements on a ratio or interval scale, whereas qualitative parameters are either ordinal or nominal. For further details please see Chapter 6.

The features of the DSI methodology are available across multiple viewpoints such as function, reliability and cost. If domain-specific knowledge can be made available in a catalogue (perhaps from a human expert or from an earlier application of the methodology), then DSI is able to automatically anticipate and populate viewpoints other than the one immediately modelled by the designer. The methodology supports the designer substituting one device for another, in which case the appropriate attributes of the new device are replaced in all viewpoints. Quantitative and qualitative calculations may be given limits, which if exceeded will raise an alarm for the designer. In this way the effects of a design change on a remote viewpoint may be assessed.

The philosophy behind the DSI methodology is that design parameters and knowledge are often uncertain rather than deterministic and mathematically explicit respectively. The DSI methodology has therefore been designed with strong capabilities for dealing with various forms of uncertainty. This has two benefits, the first that DSI may be used where information is sparse, such as in the early design stages. The second benefit is that DSI forces the user to acknowledge the existence of uncertainty and to document beliefs in a structured manner.

What is the contribution made by the software?

The DSI concepts, such as the need to consider multiple viewpoints and accommodate uncertainty, are simple enough. However to implement them requires probabilistic computation which would be prohibitively tedious for a human with pen and paper. The manual approach also provides no easy way to support the creation of multiple viewpoints. Realistically the methodology has to be embodied in a tool that supports the methodology and makes it usable. The software implementation provides this, and therefore makes an essential contribution. It supports the uncertainty processes, provides a graphical user interface, permits multiple viewpoints to be modelled, and stores the catalogue of expert knowledge or past experience. It potentially eases the cognitive load on the user as it can provide default values. It is inconceivable that anyone would use the DSI methodology without this or a similar software tool. The methodology itself is a philosophical theory

of knowledge describing the processes to achieve integrity of design. These processes are simple in concept but complex to implement in a tool.

Design functions NOT performed by DSI

To dispel any confusion, the DSI methodology does not define the problem (cf QFD) nor is it an inventive mechanism (cf decomposition methods, and TRIZ). It incorporates some of the functionality of simulation tools such as Monte Carlo analysis, Fuzzy theory cut sets, and Decision tables. It is not a system that makes decisions (cf expert systems, genetic algorithms), but it may be used to assess the risk in solutions. It is only a passive de-biassing tool as it encourages the user to admit the existence of uncertainty and to express beliefs, but it is not a critiquing system (cf Silverman 1994). It is not a method to actively record or retrieve design intent.

What kind of analysis problems can DSI cope with?

DSI requires that the solution be expressible as an acyclic graph. The graph may include quantitative and qualitative variables, and mathematical expressions alongside subjective decision maps. However the computation proceeds only once through the graph (hence acyclic).

The graph needs to be explicit as to how the outcomes depend on the inputs. The methodology provides probabilistic computation when the problem can be explicitly expressed as a mathematical equation, or as a subjective decision map. DSI has not been equipped with algorithms for solving simultaneous equations, or numerical solution methods, or mathematical transformation approaches (cf the algebra of random variables). Thus dynamic (time variant)[179] and non-linear[180] analyses will not

[179] *Time state:* Analysis may be steady state (time invariant), dynamic, or discrete events. The dynamic problems may either be concerned with equilibrium conditions (eg vibration), or transient behaviour (eg system response to perturbation). In both cases the variables are continuous and partial differential equations describe the relationships. The discrete random events include queue analysis and other complex networks.

[180] *Linearity*: Analysis may be based on variables that are either linear (eg spring characteristics) or non-linear (eg plastic deformation). Problems with linear variables are easier to solve, viz linear programming.

be immediately tractable with DSI. However it is possible that DSI could in the future be equipped to solve some such problems.[181]

17.3 **Information quality for management of design**

The previous section positioned the DSI methodology among the assessment methodologies, and described which types of analyses it supports. The discussion now shifts to consider the design decision process where one solution is selected above others, the type of information that might be required to support that decision process, and to what extent the methodology provides that information.

The design manager has to make decisions on technical content and resources committed to the project. Multiple concepts need to be explored, and one concept may be developed to greater physical detail than another. A mixed field of abstract and developed concepts makes for a difficult design decision. Yet a decision is frequently necessary in reality, even if only because there are finite design resources available. One design path will be selected for further development, usually at the expense of others. This decision making process appears to be poorly supported by formal design or decision tools. Design decisions are therefore largely made by humans using a judgement process, which is strong in that an expert human can make design decisions even when the alternatives are at very different degrees of abstraction, and weak in being vulnerable to bias.

To assess multiple concepts at different levels of abstraction it is advantageous to be able to anticipate outcomes, both good and bad. Furthermore, since decisions are generally taken under conditions of uncertainty, it is useful to know how likely those outcomes may be.

[181] Numerical solution of deterministic equations would be the first task. Solving for probabilistic variables is significantly more complex unless a single statistic (eg the mean) is used as in PERT analysis.

An assessment mechanism, of which DSI is one, needs to be able to provide quality information on which to base decisions. The quality of information was discussed in Chapter 1 and the DSI methodology may now be compared to this classification.

For those analysis cases where the methodology operates (i.e. where the output variable may be expressed as an explicit equation) DSI is able to produce both quantitative and qualitative information (cf only quantitative for Monte Carlo), full probability distributions, and multiple viewpoints. It is even able to anticipate some viewpoints, subject to a suitable catalogue being available. The DSI methodology does not address the team factor of consistency and completeness of belief, nor does it provide a conflict resolution mechanism.[182]

To summarise, the DSI methodology enables solution concepts to be modelled more richly than many other methodologies in that information can be produced that accommodates different degrees of abstraction and determinism, and from multiple viewpoints.

17.4 **Summary of capabilities of the DSI methodology**

DSI provides a probabilistic computation method that is consistent with the algebra of random variables and the probabilistic features of Monte Carlo analysis. It includes the cut set approach of Fuzzy theory, and the functionality of decision tables. The DSI method is able to process deterministic calculations too.

The literature review (Chapter 2) established that a methodology for early design should be able to (1) support multiple viewpoints, (2) compute with process variability (deterministic/crisp vs probabilistic computation), (3) process variables of different abstraction (quantitative scale vs qualitative scale), and (4) support uncertainty of

[182]The team considerations are potentially important for design. Developments in this area (see Chapter 2) are at relatively early stages, and have not yet appeared in mainstream engineering practice to any significant extent.

analysis (incompleteness of knowledge). It has been shown that DSI provides assistance in all these areas, in an integrated manner.

17.5 **Future research possibilities**

A number of directions for future research are suggested. One of these is anticipation of constraints, and as it is a larger topic it is discussed in a section on its own, followed by other briefer suggestions.

17.5.1 **Anticipating constraints in design**

Design involves both solution and *constraint* generation, but the latter is under-represented in the design literature. The 'Structured planning' approach (Owen, 1993) specifically includes constraints of various strengths,[183] but most of the effort that has gone into automating and supporting the design process, including artificial intelligence, has focussed on solution creation. The position with most design tools seems to be that if there is an intractable design problem, then what is needed is an innovative and novel solution. The corollary to this is that design impasses are caused by a designer who is ignorant of possible solution principles. Sometimes this position is valid, but there are two problems with this approach.

First, providing the explicit constraints without resorting to judgement is not a trivial undertaking in real design. Constraints have to be elucidated from vague specifications and using an element of professional judgement and experience on the part of the designer. Partly this is due to the vagueness of the specification, for example product cost may be given as an inequality ('less than $500'). Also, some parameters may be difficult to pin down even to a range, product styling being an example. Furthermore, many of the hard facts that the designer needs are not

[183]Owen (1993) distinguishes three degrees of 'defining statement', namely 'constraints', 'objectives' and 'directives' in decreasing order of compulsion. They correspond to 'must', should' and 'ought to' respectively.

contained in the specification, but have to be inferred from it. Methods like expert systems, genetic algorithms, TRIZ, qualitative simulation, and optimisation all assume that if someone will provide the constraints then the system will find a solution. All the systems can show reasonable performance in a given domain if supplied with complete problem and constraint definition. However they suffer the consequence of being tightly focussed on the domain for which the constraints can be specified. In those areas where constraints are qualitative the existing design automation systems find the terrain difficult.

The second problem is that the design space is often over-constrained, so that no solution can be found unless some constraints are relaxed. The automated design tools including artificial intelligence find this difficult to cope with. The possibility of over-constraint is not always acknowledged, for example Kuipers (1994, p6) states that ‘if all behaviours satisfy the specification, any fully specified instance of the current design will be acceptable’. However it may be impossible to satisfy every aspect of the specification, eg style, performance, reliability, noise and cost. Instead the designers seek to find constraints that may be relaxed so that a solution becomes possible. Some of these constraints are a consequence of the specification. Others are from concurrent engineering activities. Between themselves the designers negotiate and resolve the concurrent engineering requirements. Between the stakeholders (eg marketing) and the designers a lively negotiation develops as to which of the specifications are flexible enough to offer some relief. Reciprocal concessions may provide a solution, or a decision is made to freeze one design feature and make everything else compensate. At other times some of the functionality of the product will be compromised, either not being provided at all, or only while the product is in the new state (reliability compromised for the benefit of short term function). The negotiating process that occurs around constraints often means that different companies will relax different constraints, and therefore create slightly different characteristics in their products.

It may be that the real issue in engineering design is not so much solution creation but constraint anticipation. Perhaps anyone can design, but only an expert can

identify constraints where none were explicit. If anything the novice designer is not so much lacking in ideas but lacking in awareness of the constraints on success. Anticipating the constraints, especially at early design when the concept cannot be tested, may be more difficult than generating a solution. Human designers anticipate constraints on the basis of incomplete qualitative and quantitative knowledge, incorporating experience and professional judgement. They do not always get it right, as shown in the lack of robustness in many solutions.

Creating tools that help the designer *anticipate constraints* at early design would be valuable. It would likely enhance the effectiveness *solution generation* systems. The methodology might also address the issue of recording and retrieving the *design intent*, which is another constraint but a relatively little researched area of design science. Perhaps artificial intelligence tools could be redeployed to explore this aspect of design.

17.5.2 **Other potential future research**

Exploration of warranty risk

A simplified analysis was presented in this work for the warranty exposure problem, since the Weibull parameters β and η (with their variabilities) were assumed to be independent. There may be merit in attempting to develop a method to extract joint probability distributions for the Weibull parameters from raw failure data. If this could be achieved, then DSI (or an extension of it) could be used to determine warranty exposure in a more robust fashion.

Network problems

It is possible that DSI could be applied to certain network problems, particularly queueing theory, where probability distributions are propagated through a system. This might be the subject of additional research. It may be necessary to extend the software to deal with these cases.

Solution generation

The utility of existing automatic solution generation systems is uncertain at best. If an improved solution generating system was required for engineering design, then perhaps one of the emerging decision analytic expert systems (Silverman, 1994) could be suitable. Neural networks are possibly another candidate automatic solution generation system. Their benefit is that they do not need to know how the model works, they are simply trained on problems and known solutions. There may be future research possibilities in combining the probabilistic computation and multiple viewpoint features of DSI with one of these solution generation systems.

Additional validation of dishwasher wash performance model

The wash model for dishwashers could be worth studying further. In particular it would be interesting to validate the model by comparing to (a) more wash tests, and (b) more tests from other dishwashers. In addition it might be useful to develop a model to simulate glass and cutlery wash scores, as the current model only simulates crockery scores. A co-operative project with a dishwasher manufacturer could be a useful approach.

Relevance and ease of use

The DSI methodology created here requires a probabilistic approach to uncertainty. It remains to be seen to what extent designers are able to engage with these probability concepts. The risk is that the effort required to create the model may be greater than the utility of the results. The engagement effort has been a persistent problem with all automated design tools, artificial intelligence and risk assessment. DSI is a tool to express system uncertainty and manage resources so as to reduce those risks to tolerable levels. Therefore it is not only the designer's utility from the methodology that could be explored, but also that of the managers.

Enhancement of automatic viewpoint creation

The system already has the capability to automatically create other viewpoints based on the device selected from a catalogue. This could be extended to full automatic creation of relationships in those views. For example cost is mostly the summation of

all device costs, and reliability is determined from the first device to fail. It could be that a sufficiently intelligent system could recognise this and automate the relationship structure in appropriate viewpoints.

Automatic determination of consequences

The current DSI methodology is at least partially effective at determining the effect of a parameter change in other viewpoints. It does this by permitting variables to appear in multiple viewpoints, and grouping variables in the catalogue so that substitution affects all variables in the group. It is able to calculate other viewpoints and check alarm limits. It could be possible to enhance the ability of the DSI software to anticipate consequences in other viewpoints. It would be relatively simple to have the software recalculate every viewpoint whenever any parameter was changed. This is not currently implemented as it is computationally demanding and would slow the application excessively. However computer resources are likely to grow, so that this might be a practical solution in the future. Another alternative could be to create a parallel process (*thread*) to handle the recalculation of the other viewpoints in the background. Yet another alternative might be to recalculate the other viewpoints with a coarse resolution, or using a coarser method like Fuzzy theory, so that the computation was quicker.

Subjective probability

Decision analysis, forecasting, risk assessment, reliability, and operations analysis all require that the problem be defined numerically and mathematically. Though DSI provides decision maps so that mathematical relationships are non-compulsory, it still requires that probability be expressed numerically, and this could be problematic in cases where people are unable or uninterested in quantifying probability. One solution could be a modelling system that could operate on subjective probability (likely..unlikely). The work of Clarkson and Hamilton (2000) would be partially relevant to this task. As these terms can be expected to vary considerably in meaning between people, it may be necessary to provide such a modelling system with a mechanism to calibrate the terms.

De-biassing tools

An alternative approach where people are unable or uninterested in quantifying probability might instead be to prompt them to make more rational decisions, eg by showing them their biases. There has been some work in this area (Silverman, 1994) using cognitive interaction in the form of expert systems for critiquing. Ideally a expert system would quietly observe a designer and provide a critique when prompted or when bias appeared. In practice implementing such a system is difficult given the diverse tasks and thought processes that a designer undertakes. Forcing a designer to semi-continuously document all his thoughts into an expert system could significantly alter the design process, even assuming that an expert system could have sufficient natural language capabilities that the nuances of the problem could be adequately captured.

17.6 **Summary**

The DSI methodology sees design as primarily a process of reducing uncertainties, decreasing the risk, and increasing the integrity of the designed product, all from multiple viewpoints. It supports the concept of designing for key characteristics, by partially automating the viewpoint creation process and providing a catalogue. The method may start at any level with the design process, whether big picture or detail. The method is comfortable with the sparse data and large uncertainties that exist at early design. It is precisely in its handling of these uncertainties that the method differs from other approaches to managing design.

While the DSI methodology was conceived as an approach to design assessment, it is applicable to other domains that share the need to model uncertainty, such as financial modelling. DSI it is a generic tool to analyse uncertainty and support decision making even if the applications shown here have been for engineering.

Chapter 18

Conclusions

18.1 **Project achievements**

The objective of this project was to explore whether a methodology can be developed for simulating and assessing design integrity at early stages, processing both qualitative and quantitative information, accommodating uncertainty, and modelling the performance of the system from multiple viewpoints.

The results presented in the previous chapters demonstrate the development of such a methodology, its embodiment in a software system, and its application to simulate behaviour such as wash performance of a dishwasher, where relationships are qualitative and otherwise difficult to process. The same tool can also simulate performance in other viewpoints, such as cost, reliability and safety, whether those viewpoints are qualitative or quantitative or a mixture of the two. The methodology is flexible to where it is used, so it can operate at early or other stages of design, and indeed in any risk assessment process.

Ability to process uncertainty

Probability is used throughout the methodology, so that risk, uncertainty, and integrity may be assessed and used to manage the design project. By nature of its ability to process both uncertainty of analysis and process variability, the methodology accommodates the uncertainty of early design where relationships may be qualitative and variables uncertain.

Evaluation of candidate solutions

The system supports the substitution of alternative devices from a catalogue. The probabilistic algorithms also permits alternative solutions or physical devices to be evaluated, even when the precise performance of those devices is unknown. Such evaluation extends to all the viewpoints in which the design is modelled, so that interactions may be explored and change propagated through the design.

Aspects of the current project that contribute to its uniqueness are that it provides:

- engineering simulation of design (functional modelling) from more than one viewpoint,
- incorporation of cost viewpoint to functional modelling,
- modelling of engineering reliability,
- a novel model for simulating wash performance of dishwashers,
- a novel model for simulating electric shock hazard of dishwashers,
- use of probability distributions for all parameters in the simulation,
- accommodates qualitative and quantitative data,
- copes with uncertain relationships,
- creates other views as a model is developed, thereby providing a catalogue function (re-substitution from the catalogue is supported).

The methodology therefore assists the decision making process and the management of design project by providing a mechanism to model multiple viewpoints of design under conditions of uncertainty, and to determine the integrity or risk of the design.

18.2 **Closure**

The challenge for the methodologies that aim to support the design process is that design involves creative solution generation as well as the prudent anticipation of present and future constraints, with all of this being done under conditions of uncertainty. Mechanisms, including the Design for System Integrity (DSI) methodology presented here, have been developed to assist various parts of the design process. It seems clear enough that while these various mechanisms have some (and differing) ability to support the designer, they are not going to be able to displace the human designer from the core of the design process as some of the artificial intelligence projects might earlier have hoped. The human designer with creativity and experience is a wonderful designing ‘machine’. The capability of the

human mind to work with the ill-defined and even contradictory data of early design is remarkable and it seems more sensible to develop methodologies and tools to support this process rather than supplant it. Having design methodologies accessible rather than abstract, and easy to use is therefore a crucial factor in supporting the design process.

Appendix 1

Operation of the DSI software

This chapter provides details on the operation of the Design for System Integrity (DSI) software for creating probabilistic models.

A1.1 **Introduction**

Developing a probabilistic computation model for early design requires some specific modelling techniques, and these are discussed here, along with the methods that the Design for System Integrity (DSI) software requires for entering the model.

A1.2 **Selecting a distribution**

Each variable in a risk assessment model needs to be modelled with a probability distribution. There are two broad classes of distributions:

(a) Parametric distributions

These are probability distributions that are described by parameters and a mathematical relationship. Examples are the Normal, Exponential, and Weibull, among others. Such distributions may be used if there are valid grounds for believing that the relationship is true for the variable being modelled. For example, many natural phenomena have been found to follow the Normal distribution, and many life and reliability variables have been found with a Weibull distribution. A parametric distribution is used by fitting it to observed data and determining the parameters for that distribution (eg mean and standard deviation for Normal). Thereafter the parametric distribution is used instead of the raw data. Vose (1996) also feels that it is acceptable to use parametric distributions where the distribution fits the expert opinion and a low accuracy of result is acceptable, even if there no other reason to suggest a parametric distribution.

(b) Non-parametric distributions. The non-parametric distributions include the uniform, triangle and discrete distributions. These are described by their geometry. They are often easier to interface to expert opinion, (the triangle distribution is frequently used for this) but they seldom have underlying natural phenomena for their justification.

Selecting input probability distributions

When data already exist, then a probability distribution may be fitted and used as input to a simulation system. The first step is to select the family of distribution based on the shape of the density data. It is also useful if there is prior knowledge about whether data are bounded in the tails. Certain statistics such as the coefficient of variation, skewness (and others) may also point to a certain underlying distribution (Law and Kelton, 1991). After a family of distributions is hypothesised, then parameters for that distribution may be calculated from the data. Finally a test such as Chi Square may be applied to check the goodness of fit.

When data do not exist, then it is obviously more difficult to assign a type of probability distribution and its parameters. One solution is to use a triangular distribution given by the minimum, most likely and maximum values as seen by experts. However determining the absolute minimum and maximum is difficult, so using the 0.05 and 0.95 quartiles may be effective. Alternatively, a Beta distribution may be used, as this provides flexibility of shape. Law and Kelton (1991) observe that in their experience many probability densities are skewed to the right (i.e. are bounded on the left), and that this can be modelled with the Beta distribution.

The uniform and triangle distributions are often used in risk assessment. But this is more for convenience than anything else. There is no particular theoretical basis why they should be the most appropriate distributions. Often the knowledge basis is so tenuous that no distribution is particularly indicated. A major advantage of the uniform and triangle distributions is that their geometry is easier to relate to than say the Normal distribution, so that the collection of expert opinion is easier. The parameters that drive the distribution are immediately visible on the uniform and triangle distributions, which is not so for say the Normal.

The triangle distribution has the disadvantage that it has no extended lower or upper tails, but ends at well-defined points. Likewise the uniform distribution. Rare events are therefore not covered. This means that the simulation results are dependent on the lower and upper limits that the user selects for the triangle distribution. With the

triangle distribution it is unclear whether the “minimum” is the absolute minimum that could ever be conceived, or some sort of “practical” minimum. Unfortunately the results are affected one way or another.

Similar comments apply to the Beta distribution, since it too has well-defined start and end points. However the density distribution is not angular like the triangle but curved. The distribution does have an underlying theoretical basis in predicting the number of successes out of a given number of tests. A modified Beta distribution has been used for *PERT*[184] analysis in Monte Carlo by Vose (1996).

The uniform distribution is even more abrupt than the triangle, since it is flat but stops immediately at the lower and upper limits. It is difficult to see how naturally occurring variables can follow such a distribution. However the advantage is that the uniform reflects a maximum uncertainty belief (cf maximum *entropy* formulation, Vose, 1996).

The DSI project seeks to demonstrate the principles of combined qualitative and quantitative analysis, without necessarily having an exhaustive set of distributions. The method has therefore been developed with a limited range of distributions, which include the following:

- Normal distribution is included as this is a common distribution in many domains. It is also familiar to many users, in that its mean and standard deviation are more tangible than other distributions.
- Weibull distribution is included for its widespread use in reliability studies. It is also bounded on the lower side (eg times below zero are not possible), and this makes it useful in many cases where the Normal would suggest a nonsensical negative value. For example a negative product cost is not usually possible. Some Monte Carlo methods use a truncated Normal distribution to accommodate a bounded distribution, but one would have to question the rationale behind using a Normal distribution at all under such circumstances. The Exponential distribution is included as it is a special case of the Weibull.

[184]PERT analysis in project management, used to describe the uncertainty of task duration, is modelled by the user providing minimum, most likely and maximum estimates. The most likely estimate gets fours times the weighting of the others in determining the mean.

Both the conventional two parameter and the three parameter Weibull distributions are supported (the latter has a *location parameter* that offsets the curve along the horizontal axis).

- Beta distribution is included. This does not have as strong a theoretical basis as the Normal or the Weibull. However it is useful in modelling because:
  - (i) Beta distribution is bounded on both upper and lower side.
  - (ii) Beta distribution has a wide variety of shapes, including uniform (flat), straight slope, *bath tub*, and rounded triangle.
  - (iii) Beta distribution can be modelled by three parameters, minimum, typical, and maximum, and these are parameters that are arguably the easiest for a subject specialist to comprehend without getting bogged down in the statistical detail.
  - (iv) Beta distribution is widely used in project time management, in the form of a BetaPERT distribution (Vose, 1996). This four-parameter variant of the Beta is supported in the software.
- Histogram is included. This is a list of time values with their probabilities, and can be imported as a text file. Therefore the power of a spreadsheet (or other software) may be used to generate practically any parametric distribution, or to collate empirical data for inclusion into the simulation.
- Triangular distribution, for its ease of use.
- Uniform distribution.

In principle other distributions can be added to the DSI software.

A1.3 **Multiple viewpoints**

The DSI method potentially involves the generation of several probabilistic models, one for each of the required viewpoints. In particular it may be necessary to consider performance, cost, safety and reliability at the early design stages. Each of these is a separate model.

One challenge in this type of model is how to deal with variables that appear in multiple places. For example Pump power might affect several viewpoints, such as wash performance, noise, and energy usage. In Monte Carlo simulation this is achieved by ensuring that the same random value for Pump power is used in each place. In other words a new random occurrence of Pump power is NOT generated for each usage. This may be achieved by careful attention to the model, in particular making sure that for example the same cell in the spreadsheet is used each time. In the DSI method this precaution is unnecessary. This is because the DSI method never propagates a single variable through the model as does Monte Carlo. Instead it propagates a set of variables each with its own probability, i.e. an entire distribution. In Monte Carlo simulation a variable is propagated, but without any probability attached to it, since this information is unknowable to Monte Carlo. Instead Monte Carlo works out the final probability distribution by sorting all the results into a histogram. Consequently in the DSI method a calculated variable (probability distribution) may be used wherever it is required.

A1.4 **Correlation**

A second challenge in creating a model is to incorporate linked relationships. *Correlation* refers to the linking of two variables together inside a simulation. In Monte Carlo simulation this is achieved by correlation of variables using a factor.

An example of a situation where correlation is required would be a pump that can deliver high pressure or high flow, but not both at once. Therefore, pressure and flow are correlated. In Monte Carlo simulation it is necessary to set up correlations where relationships such as these exist in the model. This ensures that if the random process selects a high pressure, then it selects a low flow. If this were not done, then unrealistic results would occur, and the simulation would be contaminated. Monte Carlo simulation uses Iman & Conover correlation (Vose, 1996), which ranks the values from the two input distributions, and links them through scores.

The correlation process used in the DSI method is somewhat different and relies on maps. These are provided as part of Integrity-T, the textual probabilistic engine, but the same mechanism also accommodates numerical data. For example if pump flow is to be correlated to pressure, then a look-up table is created as shown by the model of Figure A1.1. A dummy variable (in this case 'Pump.Power') with a single entry is set up too, since the map requires two inputs. The map is shown in Figure A1.2. In this case there is only one value of pump power (300), and it is therefore operating as a dummy variable. The numbers in the bottom right 4x4 cells give the correlation, for example a low pump pressure (25) gives a high flow (7). If there is uncertainty in the correlation then that can be expressed too, as illustrated in the columns for pump pressure 75 and 125. It provides the means to determine pump flow, given the pump pressure, and thereby provides correlation. Importantly, the map also provides the means to express confidence in that relationship by spreading the probability across multiple rows. If there was some experimental data which indicated the variability of the correlation then that could be expressed in this way. In the case of a pump the characteristic curve that relates flow to pressure has some variability about it, and this may be entered in the map.

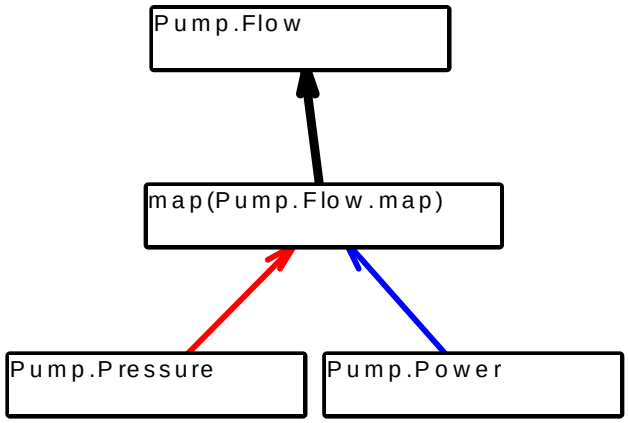


Figure A1.1: Correlation of pump flow to pump pressure using a map.

Pump.Flow			Pump.Pressure			
			25	75	125	175
PumpPower	300	Comment				
		1	0	0	0.2	1
		3	0	0.5	0.8	0
		5	0	0.5	0	0
		7	1	0	0	0

Figure A1.2: Map for determining pump flow (1, 3, 5, 7) from pump pressure (25, 75, 125, 175) for a particular value of pump power.

A1.5 **Expert opinion**

Risk assessment involves creating relationships that describe the workings of the model. Sometimes those relationships may be precise mathematical operators, for example total product cost is the sum of the costs of the sub-assemblies. However not all relationships can be expressed with such precise operators. In such cases it is necessary to resort to expert opinion (Vose, 1996).

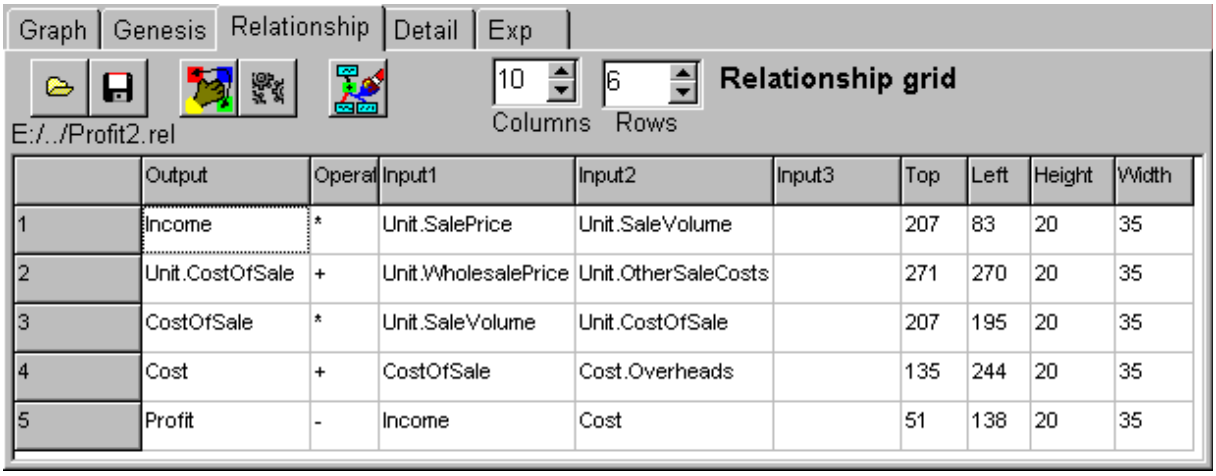
Expert opinion necessarily comes along with *bias*. Experts may be biassed in several ways, in particular by the vividness of experience (*availability bias*), and inability to conceive the range possible (*anchoring bias)*. Experts may also be affected by the organisational culture, to provide estimates that they believe comply with organisational expectations, or which advantage their personal ambitions (Vose, 1996).

A1.6 **Relationships supported in DSI**

The DSI graph is the visual representation of the relationship grid. Each line in the grid is a statement for execution.

A1.6.1 Grid structure

The structure of the relationship grid is shown in Figure A1.3 below. Each row which describes one probabilistic relationship. This includes the name of the Output and the Inputs (up to three), and the type of Operator. The four number columns at right are the screen coordinates of the operator block and are only used for the visual representation on the graph. Generally the user does not need to interact directly with the relationship grid, though it is described here for reference.



	Output	Operat	Input1	Input2	Input3	Top	Left	Height	Width
1	Income	*	Unit.SalePrice	Unit.SaleVolume		207	83	20	35
2	Unit.CostOfSale	+	Unit.WholesalePrice	Unit.OtherSaleCosts		271	270	20	35
3	CostOfSale	*	Unit.SaleVolume	Unit.CostOfSale		207	195	20	35
4	Cost	+	CostOfSale	Cost.Overheads		135	244	20	35
5	Profit	-	Income	Cost		51	138	20	35

Figure A1.3: Relationship grid.

The relationship grid is stored in a file with *.Reliability extension. The file itself is a simple text file, and may be edited with a text editor. The data is separated by commas (csv format).[185] Each row in the grid makes up one data line. Column 0 gives an index (not used), column 1 is the output filename, column 2 is the Operator, column 3 is Input1 filename, column 4 is Input2 filename, column 5 is Input3 filename or other relevant data, columns 6 to 9 give the screen co-ordinates and size of the block. It is recommended that the user put titles in row 0 (the top row), and only this row.

A mouse click on a file name loads that file into an *InspectorForm* that permits the user to view the probability distribution and values, and to calculate other statistics

[185] From version 4 all the data is stored in one *.int file.

like the mean. The *InspectorForm* also activates when a filename in the Genesis grid is clicked. If no data is shown then this could be that no file has yet been calculated, or the file is not found in the same directory as the grid.

The software supports the use of certain predefined mathematical operators. The relationships are described in the grid by special keywords.

A1.6.2 **Probabilistic Operators**

The probabilistic operators are an essential feature of the DSI method. They permit two distributions to be combined. Probabilistic operators are processed using a specially developed algorithm. This takes the discrete input distributions and determines the upper and lower limits for each interval and then the joint results. Afterwards the many combinatorial results are sorted into bins to create the output distribution.

The two and three-dimensional operators supported by the software are given in Table A1.1.

Operator between distribution D1 and D2	*Description*	*Acceptable code words in grid (May be upper or lowercase)*	*Comments*
D1 + D2		plus, sum, add, addition, +, d1+d2, d2+d1	Case 0
D1 - D2		minus, subtract, difference, d1-d2	Case 1
D2 - D1		d2-d1	Case 2.
D1 x D2		product, times, multiply, x , *, d1*d2	Case 3
D1 / D2		divide, division, d1/d2	Case 4. Beware as zero values within D2 will cause error.
D2 / D1		d2/d1	Case 5. Beware as zero values within D1 will cause error.
Min(D1, D2)		least, min, lesser, min(d1,d2)	Case 6. Use for finding earliest failure, etc.

Max(D1, D2)		larger, max, largest, max(d1,d2)	Case 7
if D1>=D2 then D2-D1 else zero		interference, interfere, lower bound, if d1>=d2 then d2-d1 else 0	Case 8. Eg Shaft-hole interference, if D1>=D2 then D2-D1 else zero
exp (- Power((TimeT/D2),D1))	D1 is the beta shape factor. D2 is eta, the characteristic life	weibull, warranty, exp(-(c/d2)^d1), exp(-power(c/d2,d1))	Case 9. Calculates reliability at a given constant time, eg the warranty time. Requires provision of time of interest, TimeT in column 5 of relationship grid.
exp(ln(c)/d1)*d2)		exp(ln(c)/d1)*d2)	51 exponential percentile life
1-d2*(1+d1)		1-d2*(1+d1)	52
discrete		discrete, discrete(d1,d2,c) discrete(d1*c,d2*(1-c))	53 discrete
D1 + D2 + D3		3daddition, 3dsum	Case 300. Three dimensional addition. Requires provision in column 5 of relationship grid of filename for probability distribution D3.
exp (- Power((D3/D2),D1))	D1 is the beta shape factor. D2 is eta, the characteristic life	3dweibull, 3dwarranty	Case 301. Calculates reliability at a given time distribution, eg the warranty time. Requires provision in column 5 of relationship grid of filename for probability distribution of time of interest. No zero values of D2 permitted.
O = D1 ^ D2		^ power, d1^d2	Case 50. Power function applied to two distributions. No complex numbers allowed, eg D1 may not be negative (unless D2 is an even integer).
R = exp(-((-(Ln(p)/D1))*D2));	D1 is the p percentile resistance. D2 is the exposure time. Both as distributions	exponentialhalflife exponentialreliability Relexp	Case 51.Reliability, using with exponential half life reliability. Output is the reliability, that is the fraction that have not yet failed at time D2, given a percentile (eg p = 0.5) resistance D1. Input D1 may not take zero value. Percentile 0<p<1.0
		1-D2-D1*D2 1-d2-(d1*d2)	52 1-D2-D1*D2

Table A1.1: Two and three dimensional operators supported by DSI.

The operators given above are not case sensitive, but they are sensitive to punctuation, including spaces. There needs to be an exact match for the system to recognise the operator. Usually several forms of an operator are provided, to

increase the robustness of the system. While some operators in the list appear in mathematical format, eg d1^d2, it is important to note that the current implementation of the software is unable to *parse* a mathematical expression to extract the sequence of operators. For example the sub expressions implicit in a statement like d1^d2*2 cannot be identified by the system. Instead the user needs present the model using operators recognised by the system. In principle it is possible to add more operators to the system, though this requires access to the source code.

A1.6.3 **Constant Operators**

The Constant operators apply a constant to a probabilistic variable. Typical examples are adding a constant to a probability distribution, or dividing a probability distribution by a constant. This is different from the probabilistic operators, in that there is only one probability distribution involved, and it is scaled in some way by the constant. Mathematical operators such as plus, divide (etc.) are used, and therefore the method is only relevant to numerical manipulations, not textual ones. The constant operator is specified in the Relationship grid (col 2) by a keyword. The constant operators in Table A1.2 are supported. Others may be added in principle.

Operator between distribution X and constant c	*Description*	*Acceptable code words in grid (May be upper or lower case)*	*Comments*
X+c	DISTRIBUTION ONE + Constant (100)	plusc, sumc, addc, additionc, +c	
X.c	DISTRIBUTION ONE x Constant (101)	productc, timesc, multiplyc, xc, *c	
X/c	DISTRIBUTION ONE / Constant (104)	dividec, divc, dividedbyc, /c	c may not be zero
c/X	Constant/DISTRIBUTION ONE (105)	cdividex, cdiv, cdividedbyx, c/x	values of X=0 will cause error
X^c	DISTRIBUTION ONE ^ Constant (110)	powerc, raisec, raisedc,^c, upc	requires positive c

Z = D1 where D1>c, else 0	Truncates distribution at constant c. Any probability in the truncated lower tail is added to the first available interval.	retainabove truncatebelow	111
Z = D1 where D1<c, else 0	Truncates distribution at constant c. Any probability in the truncated upper tail is added to the first available interval.	retainbelow truncateabove	112
		reconditionorigin, reconorigin	113
		reconditionall, reconall	114
c-X	Constant - Dist1	c-x, c-d, c-d1 among others	115

Table A1.2: The constant operators supported by DSI.

Constant operators are processed through the same probabilistic reasoning engine as 2D and 3D operators. The numerical value of the constant is to be placed by the user into the column for input distribution two.

A1.6.4 Internal processes in Numerical analysis

The following sequence of operations is provided as reference in case a user is finding it difficult to understand what the software is attempting and what input is required of the user. The sequence is as follows:

- DSI finds the operator in column 2 of the Relationship grid and identifies the type. Note that the first column of any grid is referred to as column zero in this software development.[186] Function is not case sensitive and has some limited tolerance to different expressions, eg minus and subtract.
- Software first tries Textual method before trying Numerical analysis.

[186]The first column and row in any grid or array have index zero. This is a property of the Delphi development software. The author has retained this usage into this documentation as doing so reduces the opportunity for translation error.

- System Loads the input 1 assert given in column 3. Input 1 is always a probability distribution. Software is not configured to do simple deterministic mathematics.[187]
- Checks if file exists in current directory. File name may include path if data found in another directory. Loads file into memory, or alerts user if file not found.
- Then loads the input 2 assert given in column 4. Input 2 could be a probability distribution, or a constant. The software will know its a constant by the operator. The entry in the place of file input 2 is used as the constant.
- Then initialises input array 3: Checks operator and loads file only if operator indicates need for 3rd file.
- To generate the output file, the system needs information on the preferred origin and number of steps. The software searches the Genesis grid (see next section) for the file name and if found it extracts the genesis information for this file. If the filename is not in the Genesis grid, or the data is blank, then the software uses default values. Number of output steps is expected in cell 12, and if negative, zero or blank, then the software defaults to use the setting in the Options box. If this is negative or zero then it uses 20 steps, and if the Options value is illegible then it uses the minimum number of steps in the two input distributions. The step setting is vital for the successful operation of the software, and the above safeguards ensure that a valid step number will be used despite what the operator may do. However for constant operator, the output is the same size as the input number of steps, and any number in column 12 is ignored.
- If the software is to automatically find a sensible origin and end to the output probability distribution then it needs to know what probability is considered negligible. This is called the *Resolution*. It is the probability, anything smaller than which will be ignored as far as finding the practical limits of the histogram distribution. Probability densities smaller than the Resolution are not ignored

[187]The priorities of this project were to develop a probabilistic computation engine for quantitative and qualitative simulation, and so straight deterministic mathematics has not been supported. There are many other tools that provide such capability. However this is not to say that it cannot be done, and future versions of the DSI could have such capability.

completely but will be lumped onto the upper or lower tail. Resolution is obtained from the Genesis grid, from column 9. The value depends on the number of intervals: if there are many intervals then the probability in each one will be correspondingly smaller, so a smaller Resolution will be necessary so as not to discard useful information. The user may set the Resolution directly in the column, and this may be necessary for an ill-conditioned distribution. However for most purposes the Resolution may adequately calculated automatically by the software. To activate this, simply leave the column empty in the grid, and the software then uses the setting in the Options form as follows:

Resolution := 1/(Steps0*ResolutionN)

where ResolutionN is the value from the Options Form.

- The Origin of the output file is either given by the user or calculated by the software. The software extracts the Origin from Genesis grid column 10. If blank then the system calculates the origin from the Resolution above. Usually there is no need for the user to manually set the Origin. However it is necessary if the Resolution set indirectly by the Options form is too coarse or fine for one particular probability distribution. It can also be useful to manually set the Origin (and the End, see next) if you wish to display several probability distributions on the same set of axes and have them scaled the same. This need arises only for presentation purposes, as the data is otherwise unaffected. The background mechanism is that the magnitude of a probability density depends on the interval width: fine intervals cause a lower probability density, because the distribution is a fine histogram.
- The End of the output file may be provided by the user in column 11 of the Genesis grid. If not provided then it will be calculated using the Resolution (see above). It should only be necessary to manually specify the End value under those condition described above for Origin. Note that it is not compulsory to provide an End value, even if the Origin has been specified.
- An important setting in the software is how it sorts the output results into bins. The choices are *point sort* (which lumps all probability at the centre of the interval) and *proportional sort* (which spreads the probability over the whole

interval). Generally Proportional sort is recommended, as it is very much more resistant to synchronisation artefacts, without significantly blurring of the distribution. However it is a significantly slower algorithm. The setting is in the Options form.

A1.6.5 **Textual (map) Operators**

The presence of a map is indicated in column 2, with the keyword *map(filename)*. The text in brackets is the filename of the map. When the software executes this line it will load the map and the two files named in columns 3 and 4 into the Integrity T application, and determine the output file, which it will store under the name given in column 1.

A map may be used to produce a quantitative output. This distribution may then be used in a subsequent purely arithmetic operation. To provide error-free conversion the following are recommended:

(1) The user must take care that the map has sufficient integrity. For example a map with output sequence ‘1, 2, 3 ..’ would be acceptable, but one that introduced some text strings (eg produced ‘1, k, 3, ,,’) would naturally be invalid as an interval scale and the software would raise an error if the user subsequently attempted to submit that output file to a mathematical process such as addition.

(2) The user must take care over the origin. The qualitative series does not have an origin, as each interval is fully described by a text string, and the probability is associated with this text. However a quantitative probability file does need an origin, as each point is the centre of the interval. Effectively the quantitative file format applies the stated probability frequency as a uniform distribution over each small interval, and it finds out that interval width from the origin (which is stored separately to the series itself) and the series spacing. For example the series ‘1, 2, 3..’ has an implied origin of 0.5 and the series expressed as intervals would then be ‘0.5-1.5, 1.5-2.5, 2.5-3.5....’. The user

needs to take care what origin is specified when submitting a qualitative distribution to Integrity-N, as that algorithm will assume an origin of zero if none is provided. This can sometimes cause the series to be ill-conditioned. For example if the series '1, 2, 3..' has no origin specified it will be assumed to be 0, in which case the interval widths will be alternatively long and short ('0-2, 2-2, 2-4...') and it is unlikely that this is the user's actual intent. The solution is for the user to instruct the DSI software to find the origin (using 'ReconditionOrigin' or 'ReconditionAll' as a computational step in the model) or to manually set the origin. The 'ReconditionOrigin' function simply calculates the origin based on the first few numbers in the series (eg '1, 2, 3..' has an implied origin of 0.5), which is suitable for series that are linear as x + j*y, (eg '1, 2, 3..' as well as '1, 5, 9, ..'). For all other series (such as '1, 2, 5, 10, ...') it is better to use 'ReconditionAll', as this determines an origin based on intervals at the mid points of the series. This function will also reset the centre point of each interval to make it bilaterally symmetrical.

A1.6.6 **Monte Carlo Operators**

For benchmarking purposes the DSI software also includes a Monte Carlo algorithm, see Figure A1.4 for an example of results. In this case two distributions were summed, using 1000 simulations. Monte Carlo results are characteristically jagged due to the random simulation process. The number of simulations is under the control of the user, so smoother results are obtainable at the expense of greater computation time. To use Monte Carlo, simply create a model and prefix the relationships with 'mc', as in 'mc+'. The default number of simulations is given in the Options box, though it may be overridden by entering a number for Genesis-1 (column 6) in the appropriate line of the genesis grid. The user can also enter origin,

end, and number of steps though computed defaults will be used by the software if necessary. The operators supported are shown in Table A1.3. Results from a Monte Carlo simulation may be used in a DSI model as the output file format is identical. Monte Carlo does not support textual operators, being a limitation of that method not of the algorithm used in DSI.

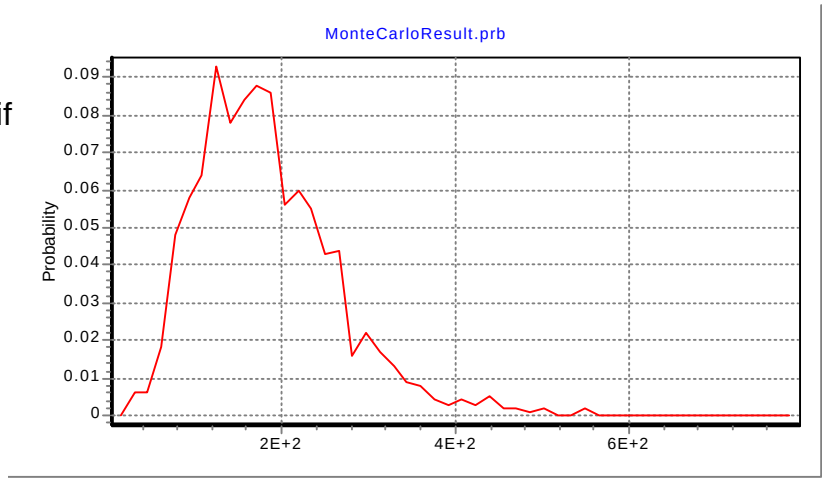


Figure A1.4: Example of Monte Carlo simulation.

Operator between distribution D1 and D2	*Description*	*Acceptable code words in grid (May be upper or lower case)*	*Comments*
D1 + D2		mc+	
D1 - D2		mc-	
D1 x D2		mcx, mc*	
D1 / D2		mc/	Beware as zero values within D2 will cause error.
D2 / D1		mcd2divided1, mcd2/d1	Beware as zero values within D1 will cause error.
Min(D1, D2)		mcleast or mcmin or mclesser mcmin(d1,d2)	Use for finding earliest failure, etc.
Max(D1, D2)		mclarger or mcmax or mclargest or mcmax(d1,d2)	
exp (- Power((TimeT/D2),D1))	Weibull where D1 is the beta shape factor. D2 is eta, the characteristic life	mcweibull or mcwarranty or mcexp(-(c/d2)^d1)') or mcexp(-power(c/d2,d1))	Calculates reliability at a given constant time, eg the warranty time. Requires provision of time of interest, TimeT in column 5 of relationship grid.
discrete		mcdiscrete or mcdiscrete(d1,d2,c) or mcdiscrete(d1*c,d2*(1-c))	

O = D1 ^ D2		mcpower or mc^ or mcd1^d2	Power function applied to two distributions. No complex numbers allowed, eg D1 may not be negative (unless D2 is an even integer).

Table A1.3: Monte Carlo operators supported by DSI.

A1.6.7 **Fuzzy theory Operators**

The DSI software also includes a fuzzy theory algorithm, which uses alpha level cut sets to compute the result of two ideas. Fuzzy theory calls its random variables ‘membership functions’ rather than ‘probability distributions’. To use the fuzzy algorithm, simply prefix the relationships with ‘fuzzy’, as in ‘fuzzy+’ for addition. When creating the model it is necessary to specifically fuzzify the inputs, which is to convert the probability distribution into a membership function. The one dimensional operator ‘fuzzify’ will do this. It is not recommended to mix fuzzy and a probabilistic computation (DSI or Monte Carlo) but if this is necessary then a ‘normalise’ operator may be used to convert a fuzzy membership into a probability distribution. A model for the product of two fuzzy triangles is shown in Figure A1.5. The model is on the left and it involves two identical inputs (triangle[0,1,2]) that are fuzzified and then multiplied together using alpha-level cuts. The result is shown in the window at right. Note that fuzzy membership functions are characterised by having a peak value of 1, whereas probability densities have an area under the curve of 1.

The default number of cut sets is given in the Options box, though it may be overridden by entering a number for Steps (column 12) in the appropriate line of the genesis grid. The origin and end are computed

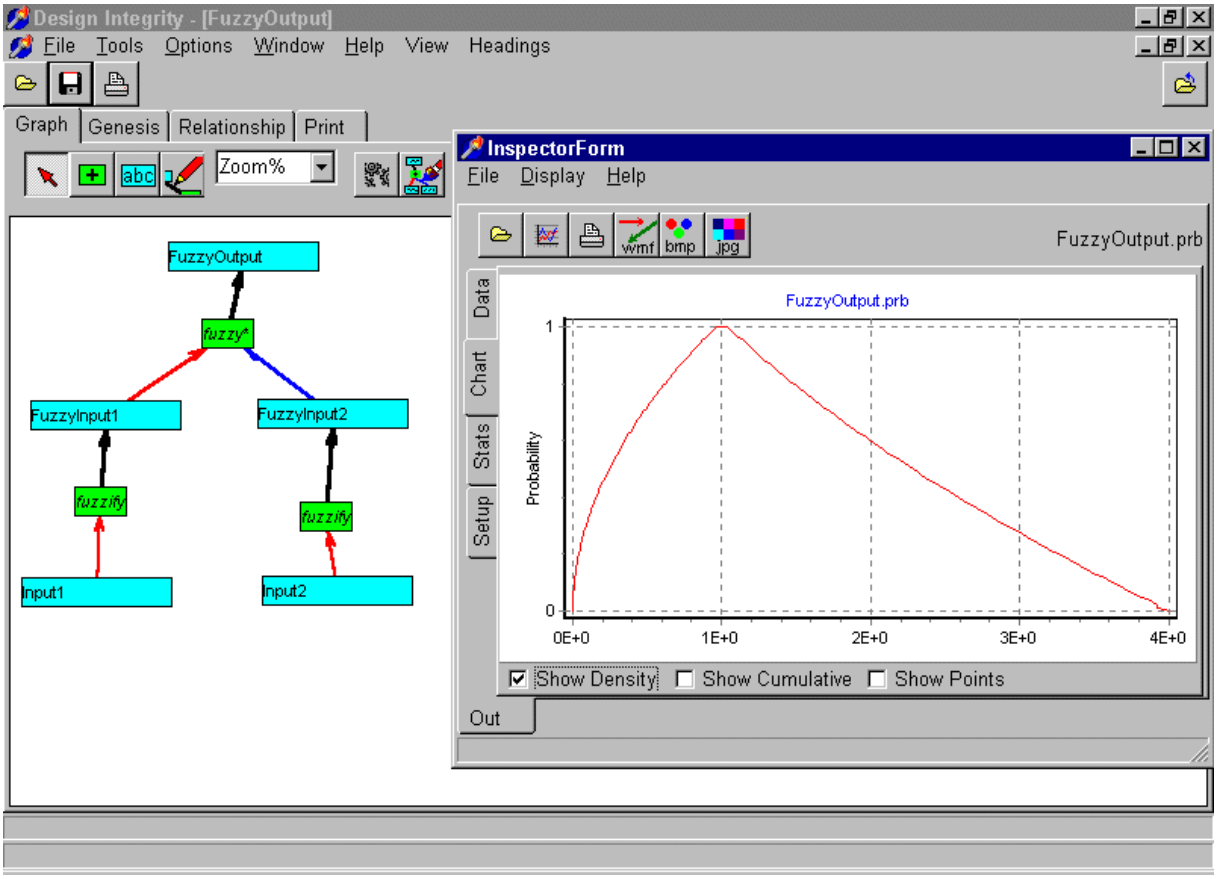


Figure A1.5: Model and result using Fuzzy theory.

from the inputs. The operators supported are shown in Table A1.4. In principle Fuzzy theory supports qualitative (textual) inputs such as *cool, warm, hot*, but only if they can be ordered and expressed on a numerical scale (eg *cool* = triangle[15,20.25]) after which algebraic operators are applied. This facility is not actively supported in the DSI software at this time. Fuzzy theory DOES NOT support qualitative operators or relationships and hence neither can such a feature be provided in the software.

Operator between distribution D1 and D2	*Description*	*Acceptable code words in grid (May be upper or lower case)*	*Comments*
D1 + D2		fuzzy+	
D1 - D2		fuzzy-	
D1 x D2		fuzzyx,fuzzy*	
D1 / D2		fuzzy/	Beware as zero values within D2 will cause error.

D2 / D1		fuzzy2/d1	Beware as zero values within D1 will cause error.
Min(D1, D2)		fuzzyleast or fuzzymin	Use for finding earliest failure, etc.
Max(D1, D2)		fuzzylarger or fuzzymax	
exp (- Power((TimeT/D2),D1))	Weibull where D1 is the beta shape factor. D2 is eta, the characteristic life	fuzzyweibull or fuzzywarranty	Calculates reliability at a given constant time, eg the warranty time. Requires provision of time of interest, TimeT in column 5 of relationship grid.
discrete		fuzzydiscrete or fuzzydiscrete(d1,d2,c)	
O = D1 ^ D2		fuzzypower or fuzzy^ or fuzzyd1^d2	Power function applied to two distributions. No complex numbers allowed, eg D1 may not be negative (unless D2 is an even integer).

Table A1.4: Fuzzy operators supported by DSI.

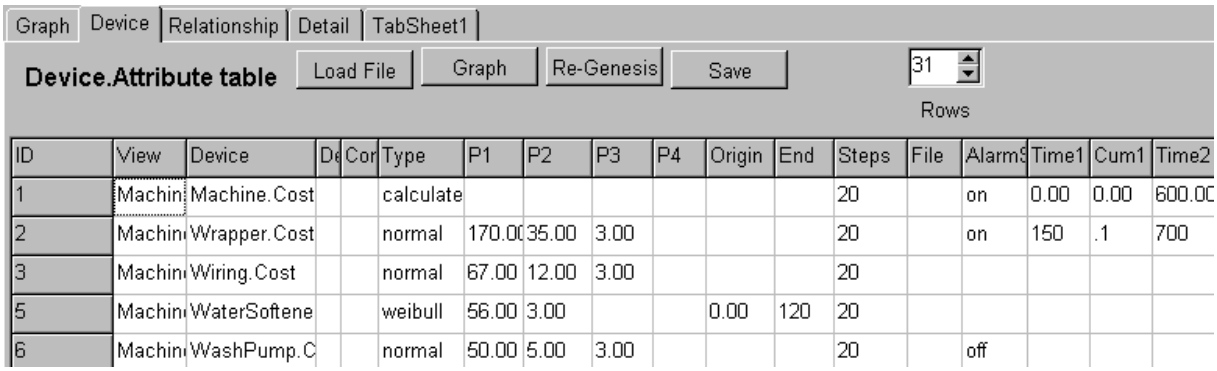
A1.7 **Genesis of distributions**

The Genesis grid contains the information necessary to create the assert files. The data are saved in comma separated variable (csv) format. Connecting to a database is possible in principle but not implemented in this edition because of the desire for portability.

A1.7.1 **Genesis grid structure**

The Genesis data are loaded into the string grid, see Figure A1.6. It describes the attributes of a device, in particular how an assert file is generated. It also gives the alarms that are set for a calculated parameters. For example the first row states that Machine.Cost is calculated (therefore no genesis parameters are given) and certain alarms are set. If the parameter exceeds these alarm specifications at run time, then the user will be notified. The name of the assert file is in column 2, and its genesis

method is in column 5. Other genesis data occupies columns 6-13. The alarms for this variable are set by parameters found in columns 18 to 23.



ID	View	Device	De	Cor	Type	P1	P2	P3	P4	Origin	End	Steps	File	Alarms	Time1	Cum1	Time2
1	Machin	Machine.Cost			calculate							20		on	0.00	0.00	600.00
2	Machin	Wrapper.Cost			normal	170.00	35.00	3.00				20		on	150	.1	700
3	Machin	Wiring.Cost			normal	67.00	12.00	3.00				20					
5	Machin	WaterSoftene			weibull	56.00	3.00			0.00	120	20					
6	Machin	WashPump.C			normal	50.00	5.00	3.00				20		off			

Figure A1.6: Genesis grid.

The statement ‘calculate’ in a genesis grid indicates that the file is calculated and not asserted. Any genesis information is therefore ignored. However the alarm information will be used during propagation. Propagation involves calculating part or all of the relationship grid, which is a separate grid. At that time the propagation software checks to see whether the variable being calculated is listed in the genesis grid, and if so it will check the alarms.

A1.7.2 **Generating the assert file**

The user may generate the assert file in one of the following ways.

A1.7.2.1 **Normal distribution**

One of the useful applications of statistics is to collect data on a sample of relatively small size, and then extend the results to the entire population. To achieve this it is necessary to have a continuous mathematical function that describes the probability of any event occurring, not discrete events or integer numbers. The continuous

probability distributions are used for this purpose, and the *normal distribution* is probably the most important of these distributions.

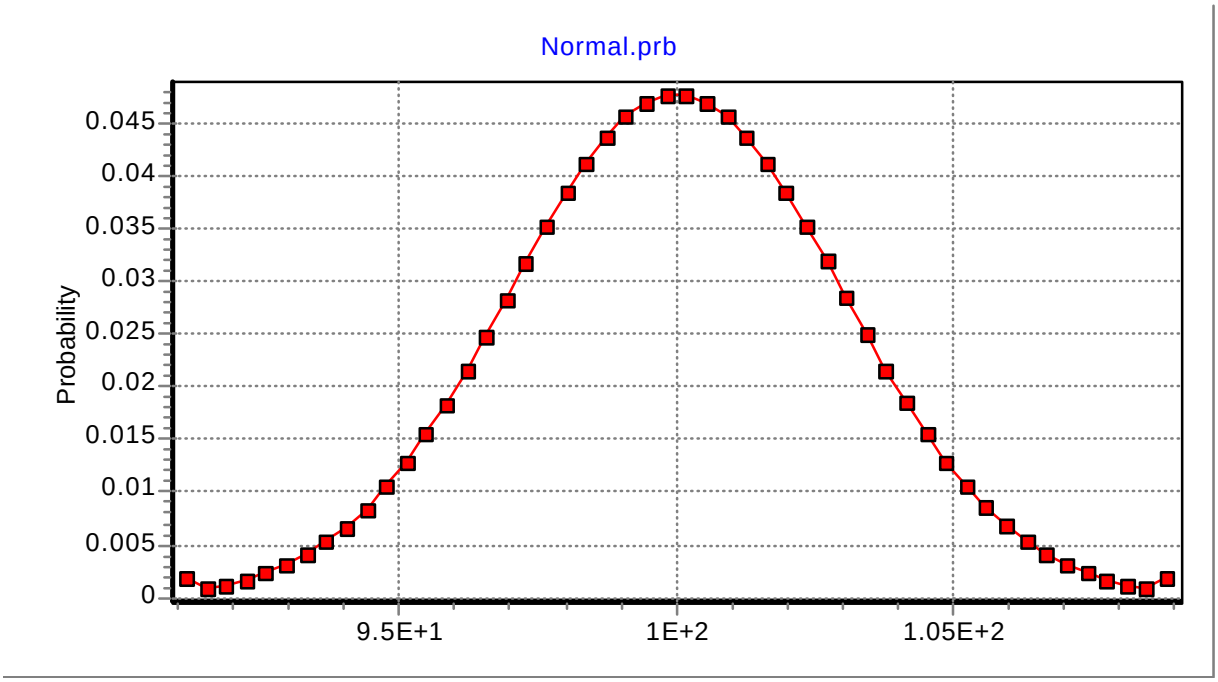


Figure A1.7: The normal distribution.

This has the classical bell shape, which indicates that most measurements will be found around the mean, and the probability of a far out event is smaller the further away it is from the mean. Figure A1.7 shows a Normal distribution modelled in DSI. It has a bell shape, and is defined by only two parameters: the mean and the standard deviation. The Normal distribution has lower and upper tails that extend indefinitely, and DSI models the distribution as a set of discrete points, and it handles the tails by truncating them at a point controlled by the user. The residual probability in the tail is then lumped onto the first (last) interval.

Mathematically the probability density function is given by

$$pd(x) = \frac{1}{s.\sqrt{(2\pi)}} \cdot \exp(-\frac{1}{2}.[\frac{(x-u)}{s}]^2)$$

where

π = 3.14159

e = 2.7183

x variable

u mean

s standard deviation

The normal curve is also called the Gaussian curve. The Normal distribution is common in statistics as it describes many natural random process. It is important to note that the Normal has lower and upper tails that extend indefinitely. This can sometimes cause difficulties in modelling work, eg if the distribution extends to negative values. If this is a problem, then rather than artificially truncate the distribution it may be worth questioning whether the Normal curve is the correct one to use. The Normal distribution may be entered in several ways. The commonest is the provision of the critical parameters (mean and standard deviation). Alternatively the user can provide several estimates to which the software fits a curve. The appropriate keywords are described below.

Normal

This statement creates a Normal distribution with the specified mean (col 6) and standard deviation (col 7). The extent of the distribution is governed by the remaining parameters. The grid is interpreted according to Table A1.5.

col 5	col 6	col 7	col 8	col 9	col 10	col 11	col 12
normal	mean	standard deviation	number of std dev on each side	step size	origin	end	number of intervals
normal	170	35	3				20

Table A1.5: Grid parameters for Normal distribution.

For example *normal, 170,35,3,0,0,0,20* creates an assert file with normal with mean 170, standard deviation 35, and the origin and end of the histogram 3 standard deviations below and 3 standard deviations above the mean, with 20 intervals. Step size will be calculated automatically.

The genesis of Normal distributions requires first the parameters of the distribution: mean (col 6) and standard deviation (col 7). Secondly it requires the user to specify how far the distribution should extent in the time axis, and thirdly how fine the steps should be.

Origin: If Origin (col 10) is provided, then software calculates using given origin, otherwise calculates origin from number of sigmas (col 8). If neither is given then execution exits.

End: The preference order for determining the end of the histogram is:

1 If Genesis End (col 11) is given then use it, providing it is not the same as the origin (exit if it is),
2 Otherwise Calculate from number of sigma (col 8).
3 Otherwise Use step size (col 9) to determine end.

If step size (i.e. interval width) is not given, then Calculate interval width as
StepWidth := (GenesisEnd-GenesisOrigin)/GenesisSteps

Calculate probability density for each interval as the upper cumulative probability less the lower cumulative probability. Use a custom Normal function for this purpose. Lower tail and upper tail are contained in the first and last density intervals respectively.

NormalFit

This fits a Normal distribution to estimates provided by the user. The user provides pairs of estimates, each pair consisting of a time value, and a cumulative probability. Typically there might be three such pairs, corresponding to estimates of minimum, expected, and maximum values. For example a cost may be ($30,10%) ($35, 50%) ($45, 98%). However longer sequences are also supported.

The normal fit will be activated if the keyword ‘normalfit’ (case insensitive) is found in column 5. The software then expects the estimates in columns 29 and onwards. The time value is required in column 29, then its probability (as a fraction of one) in

column 30, then the next time etc. Up to column 100 may be used. The programme will read estimates until it encounters a blank (‘’) cell. After the software has determined the mean and standard deviation, it will place those values back in the grid (in columns 6 and 7 respectively), and in doing so write over any values that may be there. Next the software calculates a discrete distribution using these values and the other information that the user provides in columns 8 to 12 (see normal).

NormalFitFile

If many estimates (greater than will fit up to column 100) are being used, which may arise if a previously calculated distribution is used, then it is necessary to use the keyword *normalfitfile* and provide the filename (with extension) in column 13. The data should be a text file in csv format. A probability file that has been saved from a previous operation will be valid (numerical only). Alternatively a simple csv file format will be accepted providing it has no header row, and each line consists of the time, and comma, and then the probability density.

The Normal is a symmetrical distribution and though it may be fitted to estimates that are asymmetrical, do not expect a good fit. In such cases it may be better to try the Weibull distribution, which is described below.

A1.7.2.2 **Weibull distribution**

The Weibull distribution is an extreme value distribution and is therefore often applicable in fatigue and reliability studies where failure occurs when the weakest link fails. The Weibull is bounded on the lower side, but has an infinite upper tail (see Figure A1.8). In the two parameter Weibull distribution the lower bound is at zero. The Weibull can take on various shapes, including the Exponential and an approximation to the Normal.

The density function of the Weibull distribution is

$$f(t) = \alpha\beta t^{\beta-1}.e^{-\alpha t^{\beta}} \quad or \quad = \eta^{-\beta}\beta t^{\beta-1}.e^{-(t/\eta)^{\beta}}$$

for t >0, where

α constant

β shape factor in Weibull equation

η characteristic life in Weibull equation.

The latter form with η is commonly used in reliability engineering.

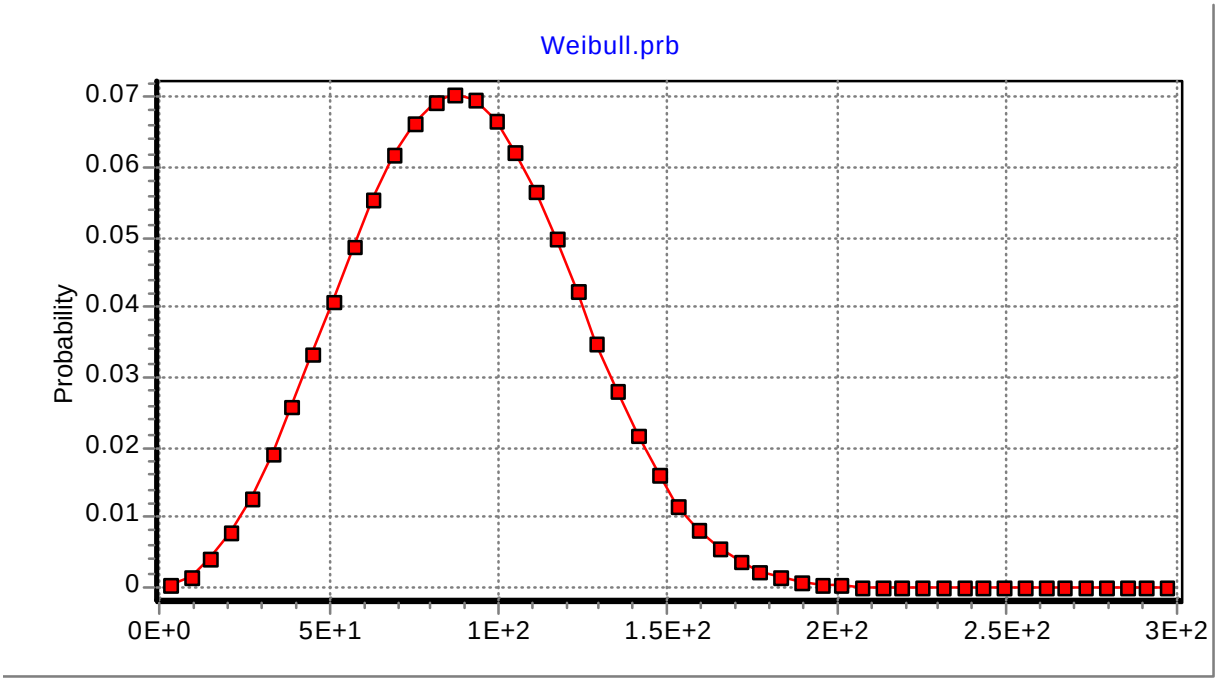


Figure A1.8: Weibull density with characteristic life 100 and shape factor 3.

With β =1 the exponential decay function is obtained, with maximum at t = 0. As β increases so the curve becomes bell shaped and moves increasingly to the right. If the failure rate decreases with time, then β <1, constant failure rate gives β = 1, and a failure rate that increases with time gives β > 1. The constant failure rate (β = 1) reduces the Weibull function to the exponential. In practice it is often assumed that the failure rate is constant (β = 1), on the grounds of simplicity or lack of other data.

The Weibull distribution F(t) is the area under the Weibull density function. In its use in reliability engineering it is given in a form with $\alpha = \eta^{-\beta}$ to give the following:

$$F(t) = 1 - e^{-(t/\eta)^{\beta}}$$

where

η characteristic life, a type of scale parameter (larger values give longer lives)

β shape factor which defines the shape of the Weibull curve and the nature of the hazard rate. It will usually be in the range from 0.5 to 4. For β < 1 the Hazard rate is decreasing and the device is getting more reliable with time. For β = 1 the Hazard rate is constant and the exponential distribution is applicable. For β > 1 the Hazard rate is increasing and the device is getting less reliable with time.

The Reliability at time t is

$$R(t) = e^{-(t/\eta)^{\beta}}$$

and the probability of failure is F(t) = 1- R(t).

The mean of the Weibull distribution is

$$\mu = \alpha^{-1/\beta}.T(1+1/\beta) = \eta.T(1+1/\beta)$$

and the variance is

$$\sigma^2 = \alpha^{-2/\beta}.[T(1+2/\beta) - [T(1+1/\beta)]^2] = \eta^2.[T(1+2/\beta) - [T(1+1/\beta)]^2]$$

where $\alpha = \eta^{-\beta}$

and the Gamma function is

$$\Gamma(\alpha) = \int_0^{\infty} x^{\alpha-1}.e^{-x}dx$$

If a mean time to failure is required then the mean may be used. However it is often better to use the median, which is the time by which half the devices have failed. The mean and the median are not the same for the Weibull distribution because the distribution is skewed.

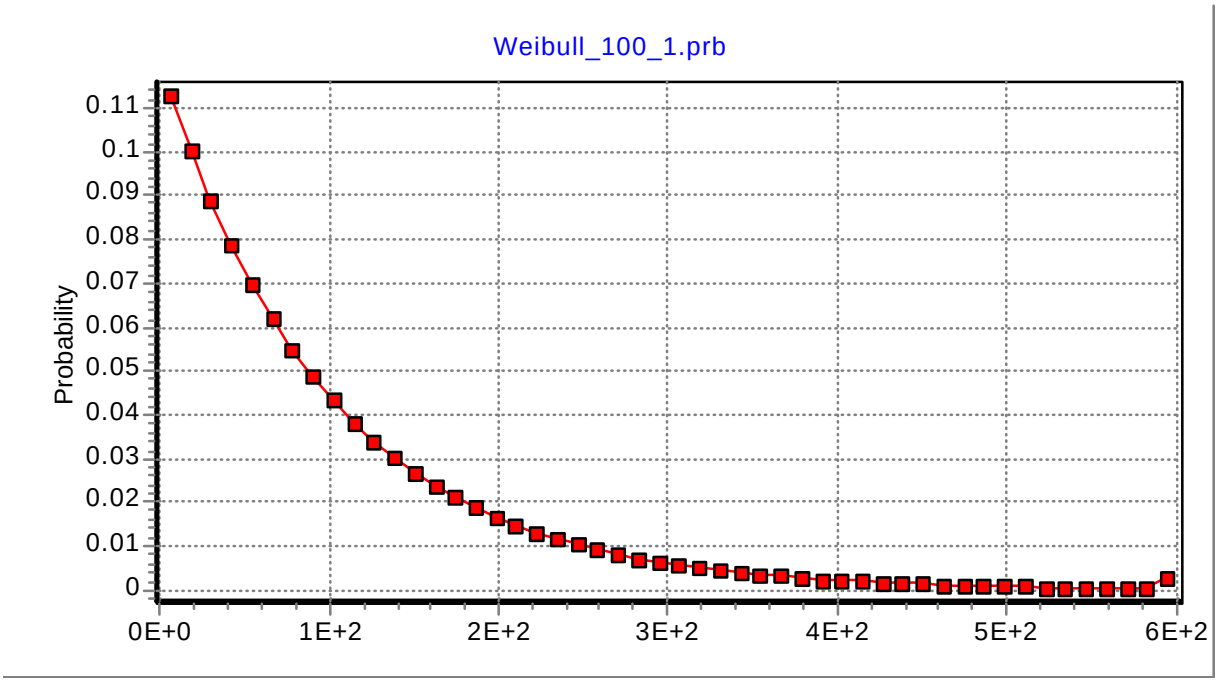


Figure A1.9: Exponential distribution obtained with Weibull characteristic life 100 and shape factor 1.

The Exponential distribution (Figure A1.9) is a subset of the Weibull. It is often used in reliability studies as it is applicable with constant failure rate. The density function is

$$f(t) = \frac{1}{\eta}.e^{-t/\eta}$$

although it is common to find the substitution $\eta = 1/\lambda$ where

λ failure rate

and is constant when the exponential distribution is used.

The Weibull distribution may be entered in several ways including provision of the critical parameters (characteristic life eta and shape factor beta) or provision of several estimates to which the software fits a curve. The appropriate keywords are described below.

Weibull

This code creates a Weibull distribution with the specified characteristic life eta (col 6) and shape factor beta (col 7). A shape factor must be greater than zero. A value of 1 gives an Exponential distribution. Values around 4 give an approximate Normal distribution. At high values (eg over 10) the Weibull distribution becomes degenerate. A *location parameter* may be provided in col 8 for a three parameter Weibull. In most cases the location parameter would be blank or zero. The extent of the distribution is governed by the remaining parameters. The grid is interpreted as per Table A1.6.

col 5	col 6	col 7	col 8	col 9	col 10	col 11	col 12
weibull	characte ristic life eta	shape factor beta	location parameter	step size	origin	end	number of intervals
weibull	2000	2	may be blank	may be blank	may be blank	may be blank	may be blank

Table A1.6: Interpretation of Weibull parameters.

Step size, Origin, End and Number of intervals are interpreted as for Normal distribution, except that number of sigmas is not applicable for Weibull.

WeibullFfit

This command in column 5 fits a Weibull distribution to estimates provided by the user. The data entry is identical to that for ‘NormalFit’ above, to which the reader is referred. The Weibull is an asymmetrical distribution and it therefore often gives a closer fit than the Normal to estimates that are asymmetrical.

WeibullFitFile

If many estimates (greater than will fit up to column 100) are being used, which may arise if a previously calculated distribution is used, then it is necessary to use the above keyword and provide the filename (with extension) in column 13. The data should be a text file in csv format. A probability file saved from a previous operation will be valid (numerical only). Alternatively a simple csv file format will be accepted providing it has no header row, and each line consists of the time, and comma, and then the probability density.

A1.7.2.3 **Beta distribution**

The Beta distribution is bounded on two sides, eg 0 and 1, or a and b. It is a versatile distribution as it can model bell shapes (Figure A1.10), bath tub, flat line (uniform), triangular (right sided), and exponential-like shapes. Its theoretical basis is that it gives the probability of the event, given that r successes have been already found out of r tests. Its use in risk modelling is usually rather for the convenience of its dual bounded nature rather than anything else.

The Beta distribution is sometimes used to model expert opinion in a PERT diagram (project management), using the min, most likely and max estimates. A factor of 4 is usually used. The Beta is a continuous distribution. The mathematical expression for the beta density distribution is

ProbBeta[i] := PowerX(Betax[i],alpha1-1)*PowerX(1-Betax[i],alpha2-1)/Area;

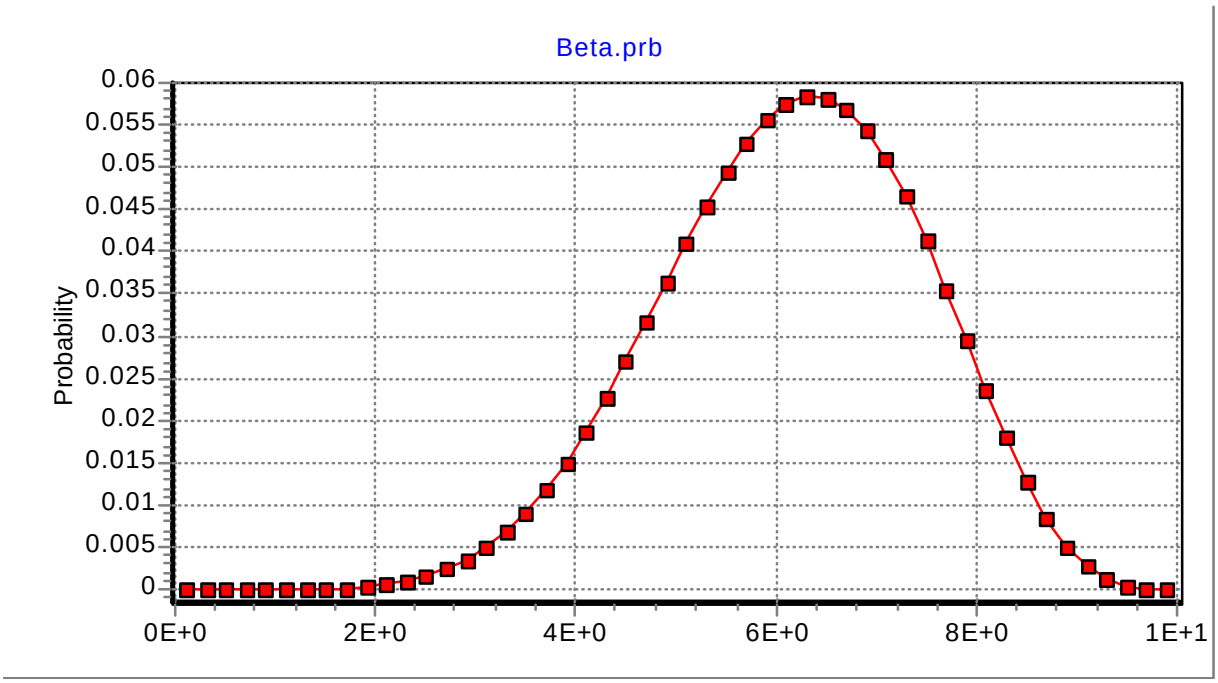


Figure A1.10: Beta distribution (8,5) giving a bell shaped curve.

DSI is equipped with a Beta distribution generator as an add-on. This is called *BetaDistribution* and it is available off the main menu under tools. This form allows the user to explore different beta distributions. Once the format has been decided, then the appropriate keyword and parameters may be inserted in the *genesis* grid.

Beta

The code word *beta* (case insensitive) will generate a beta distribution with alpha1 and alpha2 as given by the Genesis1 and Genesis2 parameters (columns 6 and 7 resp.), i.e. User provides direct input of the beta parameters.

BetaTrial

User provides parameters r successes out of n trials in the Genesis1 and Genesis2 parameters (columns 6 and 7 resp.). The system then calculates:

```
alpha1 :=r+1;
alpha2 := n-r+1;
MeanBeta := alpha1/(alpha1+alpha2);
```

BetaPERT

The User provides min, max, and most likely time value. The user also provides a factor (usually 4) to set the sharpness of the peak. The system then calculates the beta parameters as follows:

```
MeanBeta := (PertMin+PertFactor*PertLikely+PertMax)/(PertFactor+2);
alpha1 := (MeanBeta-PertMin)*(2*PertLikely-PertMin-PertMax)
/((PertLikely-MeanBeta)*(PertMax-PertMin));
alpha2 := alpha1*(PertMax-MeanBeta)/(MeanBeta-PertMin);
```

A1.7.2.4 Triangle distribution

Using the code word ‘triangle’ will activate a triangular distribution. The user provides the lower, peak and upper values in columns 6, 7 and 8 respectively. See Figure A1.11 for the results. This is a relatively simple distribution to use, and easy to connect to expert opinion. The difficulty in using expert opinion is whether the ‘lower’ value represents the absolute lowest value or a practical minimum. Similar considerations apply to the ‘upper’ value. The problem is that the triangle distribution has no tails, and the interpretation can affect the simulation results.

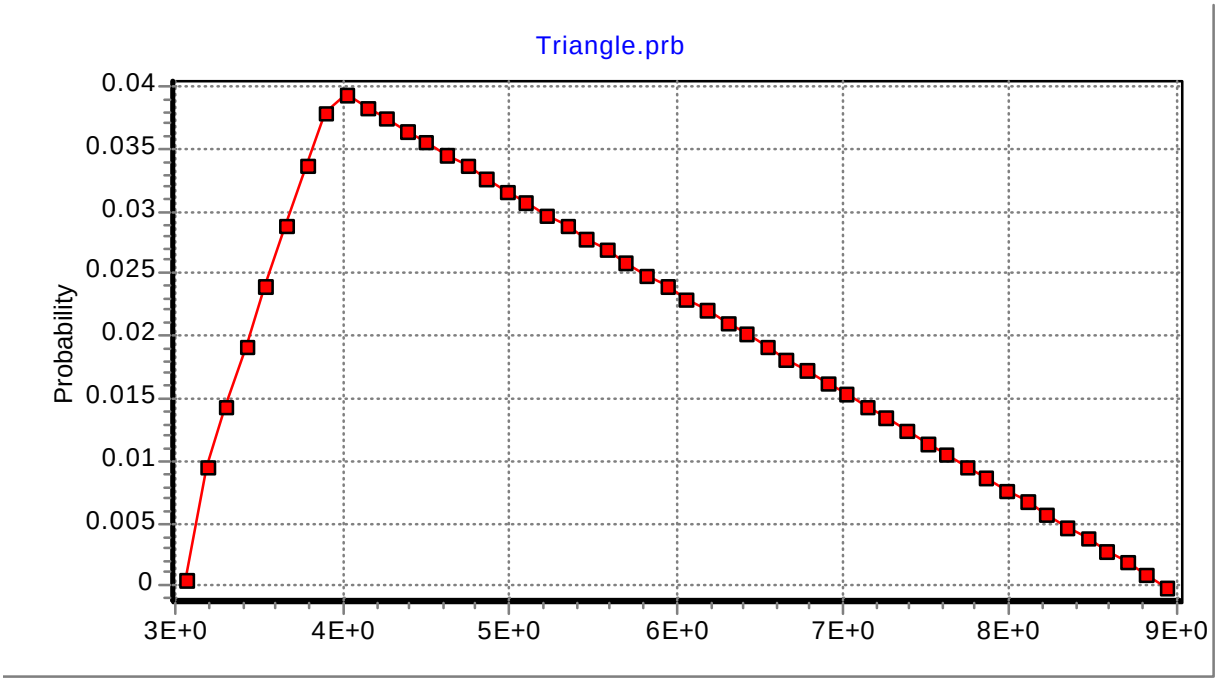


Figure A1.11: Triangular distribution (3,4,9)

A1.7.2.5 Uniform distribution

Using the code word ‘uniform’ will activate a triangular distribution. The user provides the lower and upper values in columns 6 and 7 respectively. The uniform distribution is flat between the lower and upper limits, and cuts of abruptly outside of the limits (i.e. it has no tails). It is useful when there is no opinion about the variability of the

process. However Vose (1996) believes it is rare that a person is able to define the bounds but has no opinion about the tails or central tendency. The distribution should therefore be used with care.

A1.8 **Discussion**

The DSI software has been written to support both quantitative and qualitative probabilistic computation. It provides these capabilities within a fully developed graphical user interface, to maximise its usefulness. The DSI software has been provided with many error trapping routines to ensure the integrity of the calculations, and to provide responses to the user about modelling errors. Most of the conditions that could drive the software to fatal failure are protected against, but a determined user could likely still force error conditions. The software is a specialised research tool and is possibly unsuitable for a naive user. Professional judgement and technical skill are required to ensure that those results are relevant to the problem being modelled.

The software is at the state where it may be used by people other than the author, if such users have some explanation of how to use it. This chapter has attempted to provide such an explanation. This is not an attempt at all-inclusive documentation but intends to be sufficient to get a reader familiar with probability concepts to a position of being able to use the software and test the claims being made for it.

Appendix 2

Test data

A2.1 **Chi square test on validation example**

The following data refer to the validation example of the quotient of two uniform variables. A chi square test was used to compare the DSI results to those predicted from the algebra of random variables. One hundred data points were used for each of the DSI variables. The data are shown in Table A2.1.

A	B	C	D	E
from file	from file	=+B5/(A5-A4)/2	=+IF(A5<1,0.5,1/(2*A5^2))	=+(C5-D5)^2/D5
Time	DI Histogram	DI Density	Mellin	Chi sq
0		hd(x)	h(x)	
0.02	0.019926	0.498156	0.5	6.79893E-06
0.06	0.019987	0.499664	0.5	2.26143E-07
0.1	0.019991	0.499774	0.5	1.02234E-07
0.14	0.019993	0.499817	0.5	6.68538E-08
0.18	0.019996	0.499891	0.5	2.36907E-08
0.22	0.019992	0.499803	0.5	7.7551E-08
0.26	0.019994	0.499853	0.5	4.33451E-08
0.3	0.019995	0.499882	0.5	2.80559E-08
0.34	0.019994	0.499856	0.5	4.13248E-08
0.38	0.019997	0.499913	0.5	1.49763E-08
0.42	0.019993	0.499835	0.5	5.46323E-08
0.46	0.019997	0.499921	0.5	1.25371E-08
0.5	0.019995	0.499879	0.5	2.94088E-08
0.54	0.019993	0.499831	0.5	5.73177E-08
0.58	0.019996	0.499909	0.5	1.66314E-08
0.62	0.019994	0.49984	0.5	5.12753E-08
0.66	0.019996	0.499891	0.5	2.35931E-08
0.7	0.019995	0.499869	0.5	3.41441E-08
0.74	0.019996	0.499892	0.5	2.34074E-08
0.78	0.019997	0.499924	0.5	1.14789E-08
0.82	0.019995	0.499866	0.5	3.5945E-08
0.86	0.019997	0.499914	0.5	1.46497E-08
0.9	0.019998	0.49994	0.5	7.16238E-09
0.94	0.02	0.500002	0.5	5.31114E-12
0.98	0.020031	0.500773	0.5	1.19379E-06
1.02	0.019189	0.479716	0.480584	1.56849E-06
1.06	0.017795	0.444884	0.444998	2.92282E-08
1.1	0.016529	0.41323	0.413223	1.13511E-10
1.14	0.01539	0.384752	0.384734	8.638E-10
1.18	0.014363	0.359068	0.359092	1.58695E-09
1.22	0.013435	0.335886	0.335931	6.05811E-09
1.26	0.0126	0.314994	0.314941	9.12131E-09
1.3	0.011832	0.295788	0.295858	1.6725E-08

1.34	0.011135	0.278367	0.278458	3.03491E-08
1.38	0.010501	0.262534	0.26255	9.16095E-10
1.42	0.009916	0.247899	0.247967	1.83002E-08
1.46	0.009382	0.234544	0.234566	2.06525E-09
1.5	0.008889	0.22223	0.222222	2.7328E-10
1.54	0.008432	0.2108	0.210828	3.68093E-09
1.58	0.008005	0.200126	0.200288	1.31483E-07
1.62	0.007625	0.190613	0.19052	4.58167E-08
1.66	0.007249	0.181218	0.181449	2.92237E-07
1.7	0.006922	0.173041	0.17301	5.24917E-09
1.74	0.006603	0.165066	0.165147	3.99174E-08
1.78	0.006308	0.157688	0.157808	9.24614E-08
1.82	0.006035	0.150886	0.150948	2.54534E-08
1.86	0.005781	0.144537	0.144525	1.00557E-09
1.9	0.005542	0.138538	0.138504	8.10916E-09
1.94	0.005316	0.132905	0.132852	2.12163E-08
1.98	0.005119	0.127976	0.127538	1.50331E-06
2.02	0.004878	0.121952	0.122537	2.79321E-06
2.06	0.004705	0.117625	0.117824	3.38502E-07
2.1	0.004528	0.113203	0.113379	2.71901E-07
2.14	0.004361	0.109022	0.10918	2.28054E-07
2.18	0.004204	0.105097	0.10521	1.21402E-07
2.22	0.004054	0.10136	0.101453	8.50058E-08
2.26	0.003916	0.0979	0.097893	4.07593E-10
2.3	0.003778	0.09446	0.094518	3.57761E-08
2.34	0.003644	0.091103	0.091314	4.87902E-07
2.38	0.003528	0.088208	0.088271	4.43095E-08
2.42	0.003419	0.085464	0.085377	8.96504E-08
2.46	0.003295	0.082379	0.082623	7.16426E-07
2.5	0.003197	0.079918	0.08	8.44687E-08
2.54	0.003102	0.07754	0.0775	2.0687E-08
2.58	0.002995	0.074873	0.075116	7.83603E-07
2.62	0.002915	0.072873	0.07284	1.54755E-08
2.66	0.00282	0.070491	0.070665	4.27882E-07
2.7	0.002741	0.068533	0.068587	4.28831E-08
2.74	0.002661	0.066525	0.066599	8.37241E-08
2.78	0.002584	0.064609	0.064696	1.18798E-07
2.82	0.002513	0.062824	0.062874	3.9732E-08
2.86	0.002441	0.061033	0.061128	1.45955E-07
2.9	0.002376	0.059411	0.059453	2.99906E-08
2.94	0.002314	0.057847	0.057846	8.66339E-12
2.98	0.00226	0.056499	0.056304	6.75122E-07
3.02	0.002182	0.054538	0.054822	1.46926E-06
3.06	0.002124	0.053103	0.053398	1.63778E-06
3.1	0.002084	0.052105	0.052029	1.09479E-07
3.14	0.002018	0.050442	0.050712	1.43404E-06
3.18	0.001976	0.049402	0.049444	3.54836E-08
3.22	0.001927	0.048187	0.048223	2.69162E-08

3.26	0.001873	0.046829	0.047047	1.01101E-06
3.3	0.001839	0.045971	0.045914	7.07956E-08
3.34	0.001786	0.044656	0.044821	6.0534E-07
3.38	0.001745	0.043633	0.043766	4.0451E-07
3.42	0.001713	0.042829	0.042748	1.52407E-07
3.46	0.001663	0.04157	0.041766	9.12025E-07
3.5	0.001628	0.040691	0.040816	3.86902E-07
3.54	0.001598	0.039956	0.039899	8.18883E-08
3.58	0.001554	0.038845	0.039013	7.1997E-07
3.62	0.00152	0.037992	0.038155	6.95355E-07
3.66	0.001494	0.037354	0.037326	2.12011E-08
3.7	0.001459	0.036475	0.036523	6.22561E-08
3.74	0.001421	0.035536	0.035746	1.2364E-06
3.78	0.001398	0.034954	0.034993	4.52404E-08
3.82	0.001374	0.034356	0.034264	2.44481E-07
3.86	0.001334	0.033353	0.033558	1.25134E-06
3.9	0.00131	0.032744	0.032873	5.0409E-07
3.94	0.00129	0.032239	0.032209	2.85267E-08
3.98	0.126732	3.168309	0.031565	311.7126669

Table A2.1: Data for DSI and the algebra of random variables.

Column A contains the time scale, with the centre point of the interval. This data came from the DSI file. Column B contains the probability calculated by DSI. This is histogram data and in Column C it is converted to true density by dividing by the interval width. Column D contains the density predicted by the algebra of random variables, and this is the reference data. Column E is the chi square value.

The sum of the Chi Square values in column E is 3.36707E-05. This excludes the last data point where DSI provides the residual tail but not the algebra of random variables. The degrees of freedom is then one less the number of data points: DoF = 98. Using a significance of 0.9999999 (the computational limit of MS Excel), the one tailed test statistic is 12.207. A one sided test is used as we are only interested in finding out whether or not the observed difference is larger than that due to chance.

The observed chi square total is very much less than the test statistic and therefore it is concluded that there is no significant difference between the data from the DSI and the algebra of random variables at the 99.99999% significance level.

A2.2 **Chi square test on robustness example**

These data refer to the quotient of two uniform random variables. This chi square test is for goodness of fit of DSI results with only 10 data points per variable. The reference is the result from the algebra of random variables. The data are shown in Table A2.2.

A	B	C	D	E
from file	from file	=+B5/(A5-A4)/2	=+IF(A5<1,0.5,1/(2*A5^2))	=+(C5-D5)^2/D5
Time	DI Histogram	DI Density	Mellin	Chi sq
0		hd(x)	h(x)	
0.2	0.191695804	0.47924	0.5	0.000861996
0.6	0.196721191	0.491803	0.5	0.000134382
1	0.180092666	0.450232	0.5	0.004953775
1.4	0.1019464	0.254866	0.255102	2.18402E-07
1.8	0.062496487	0.156241	0.154321	2.38936E-05
2.2	0.037737263	0.094343	0.103306	0.000777582
2.6	0.027471528	0.068679	0.073964	0.000377727
3	0.021016983	0.052542	0.055556	0.000163418
3.4	0.015111	0.037778	0.043253	0.00069306
3.8	0.165710678	0.414277	0.034626	4.162607859
Sum				**0.007986052**

DoF	8
Significance	0.999999900
Test statistic	**0.047683716**

Table A2.2: Data for DSI with ten data points, and the algebra of random variables.

Column A contains the time scale, with the centre point of the interval. This data came from the DSI file. Column B contains the probability calculated by DSI. This is histogram data and in Column C it is converted to true density by dividing by the interval width. Column D contains the density predicted by the algebra of random variables, and this is the reference data. Column E is the chi square value.

The sum of the Chi Square values in column E is shown as 0.00798. This excludes the last data point where DSI provides the residual tail but not the algebra of random variables. The degrees of freedom is then one less the number of data points: DoF =

8. Using a significance of 0.9999999 (the computational limit of MS Excel), the one tailed test statistic is 0.04768. A one sided test is used as we are only interested in finding out whether or not the observed difference is larger than that due to chance.

The observed chi square total is very much less than the test statistic and therefore it is concluded that there is no significant difference between the data from the DSI and the algebra of random variables at the 99.99999% significance level.

A2.3 **Scoring wash tests in the DSI model**

This section documents the method used to calibrate the DSI model for wash performance (see section 11.5), by following the process used on one of the sets of data. The source data is a wash test conducted to ANSI/AHAM standards. Scores were recorded (Gin, 2000) as shown in Table A2.3.

	1	2	3	4	5	6	7	8	9	10
Crockery										
Dinner Plate	3	1	6	5	5	4	8	6	6	9
Fruit Bowls	9	9	9	9	6	4	5	2	9	2
B/B Plates	9	9	3	9	9	1	3	2	6	9
Saucers	1	6	1	4	7	2	2	3	2	4
Cups	9	9	9	6	9	9	4	5	5	9
Serving Dishes	6	2	5							
Glasses	9	9	9	9	9	9	9	9	9	9
Cutlery										
Dinner Forks	4	9	4	9	8	9	3	9	7	8
Salad Forks	2	5	9	5	5	7	7	5	3	4
Knives	9	9	6	9	9	9	6	9	9	9
Teaspoons	9	6	9	8	9	9	9	9	9	9
Teaspoons	4	3	3	4	3	8	4	5	6	6
Serving utensils	9	9	5							

Table A2.3: Raw wash data scores for ASKO 1805, Detergent Amounts 15 & 30g, Programme Normal, test reference CP:11 of 15/07/1998. The numbers are the demerit points for each of up to 10 dishware items. The worst possible score in this test is 9.

The ANSI/AHAM method would be to count the occurrence of each type of fault (Table A2.4). Next the method determines the frequency, applies the weighting factors (Equation 11.1) to calculate a wash index for each type of dishware and finally determine an overall wash index weighted by the total faults (Table A2.5).

Fault type x	Crockery	Glasses	Cutlery
0	0	0	0
1	4	0	0
2	7	0	1
3	4	0	5
4	5	0	6
5	6	0	6
6	8	0	5
7	1	0	3
8	1	0	4
9	17	10	23
Sub Total	53	10	53
Total			116

Table A2.4: Count of various fault (0-9) occurrence, for different categories of ware.

Fault type x	Crockery	Glasses	Cutlery
0	0.0%	0.0%	0.0%
1	7.5%	0.0%	0.0%
2	13.2%	0.0%	1.9%
3	7.5%	0.0%	9.4%
4	9.4%	0.0%	11.3%
5	11.3%	0.0%	11.3%
6	15.1%	0.0%	9.4%
7	1.9%	0.0%	5.7%
8	1.9%	0.0%	7.5%
9	32.1%	100.0%	43.4%
Wash index	42.0%	0.0%	31.1%
Weight	0.456897	0.086207	0.456897
Overall wash index			33.4%

Table A2.5: Frequency of fault is used to determine the wash index for each ware type, and then the overall wash index.

However the DSI model of wash performance departs from the ANSI/AHAM method described above. The first departure is that it takes only the crockery scores, as (i) the glasses are suspected to be subject to other wash effects as they consistently have very much worse wash scores than other ware, and this is also evident in wash tests on other machines not shown here, and (ii) the cutlery is relatively close-packed

in a small basket and there are potentially other effects operating there too. The second departure is to rescale the scores to 0 to 10. Table A2.6 shows the results, with data from another test (CP10) on the same machine added into the table to increase the data set.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Test CP11										
Dinner Plate	3.3	1.1	6.7	5.6	5.6	4.4	8.9	6.7	6.7	10.0
Fruit Bowls	10.0	10.0	10.0	10.0	6.7	4.4	5.6	2.2	10.0	2.2
B/B Plates	10.0	10.0	3.3	10.0	10.0	1.1	3.3	2.2	6.7	10.0
Saucers	1.1	6.7	1.1	4.4	7.8	2.2	2.2	3.3	2.2	4.4
Cups	10.0	10.0	10.0	6.7	10.0	10.0	4.4	5.6	5.6	10.0
Serving Dishes	6.7	2.2	5.6							
Test CP10										
Dinner Plate	10.0	7.8	5.6	5.6	2.2	6.7	10.0	10.0	8.9	4.4
Fruit Bowls	10.0	10.0	10.0	10.0	1.1	2.2	10.0	3.3	7.8	2.2
B/B Plates	4.4	7.8	4.4	5.6	4.4	5.6	8.9	3.3	5.6	10.0
Saucers	4.4	4.4	4.4	2.2	2.2	6.7	3.3	1.1	3.3	2.2
Cups	10.0	5.6	10.0	10.0	6.7	4.4	7.8	10.0	10.0	5.6
Serving Dishes	6.7	5.6	3.3							

Table A2.6: Data for two wash tests is shown here, with only the scores for crockery included. Scores have been stretched to a 0 to 10 scale on a proportional basis.

The wash performance used in this model is simply the mean score from the table. This is a simple calculation to perform and the result (using MicroSoft Excel) is mean wash performance 38.05% (and standard deviation 30.32). This result differs slightly from the 42% wash index calculated for crockery by the ANSI/AHAM method above. The standard deviation must be interpreted with care, and as a measure of dispersion and not as a characteristic of a Normal distribution since the distribution is truncated at 0% and 100%.

Importantly, the wash model used here makes visible the uncertainty in the result, since the fault data may be counted (Table A2.7) and used to determine the frequency of wash performance (Table A2.8). The final result is the histogram shown (above) in Figure 11.12.

Demerit points	Count
10	31
8.89	3
7.78	5
6.67	12
5.56	14
4.44	13
3.33	9
2.22	13
1.11	6
0.00	0

Table A2.7: Count of various fault scores, based on the data in the previous table.

Wash percent	Frequency
0.00	0.292453
11.11	0.028302
22.22	0.04717
33.33	0.113208
44.44	0.132075
55.56	0.122642
66.67	0.084906
77.78	0.122642
88.89	0.056604
100.00	0

*Table A2.8: Data for wash performance histogram. Wash percent is determined from the demerit point score as (10-score)*10. Frequency is determined from the Count column of the previous table, divided by the total number of faults.*

The above data are for one particular model of machine. The process may be repeated for test data from other dishwashers.

A2.4 **Chi square test for goodness of fit wash percent**

This section documents how Chi square goodness of fits tests were conducted to compare the results from the DSI wash performance simulation with the observed data for an ASKO 1805 dishwasher (see section 11.5). The wash performance results predicted by the simulation are shown in Table A2.9.

Wash performance	Frequency
1	0.00243
3	0.23145
5	0.010381
7	0.010861
9	0.011358
11	0.011869
13	0.012393
15	0.012931
17	0.013482
19	0.014044
21	0.014614
23	0.015192
25	0.015776
27	0.016367
29	0.01696
31	0.017549
33	0.018133
35	0.01871
37	0.019276
39	0.019827
41	0.020357
43	0.02086
45	0.021334
47	0.021768
49	0.022156
51	0.022518
53	0.022783
55	0.022977
57	0.023093
59	0.023121
61	0.023045
63	0.022856
65	0.022546
67	0.022103
69	0.021514
71	0.020766
73	0.019851

75	0.018759
77	0.017483
79	0.016022
81	0.014379
83	0.012568
85	0.010617
87	0.008573
89	0.006513
91	0.004541
93	0.002791
95	0.001416
97	0.000664
99	6.31E-06

Table A2.9: Data for simulated wash performance. The first column is the wash performance [%] and the second column is the frequency. This data is the tabular version of the graph shown previously in Figure 11.13.

The data were sorted into the wider bins corresponding to the observed data (see previous Table A2.8) so that comparison could be made. The actual occurrence was known (see Table A2.7). The simulated occurrence and Chi square value (being a measure of the difference) was determined as per Table A2.10. Occurrences of 5 or less were merged into an adjacent bin.

Wash performance bins Bin sizes Start	**Mid**	End	Actual occurrence A	Simulated Frequency Sf	Simulated occurrence S	Chi square C = (A-S)^2/A
-	**-**	5.56	31	0.244	25.89	0.0964
5.56	**11.11**	16.67	3	0.059	6.30	
16.67	**22.22**	27.78	5	0.089	9.48	
27.78	**33.33**	38.89	12	0.091	9.61	0.2572
38.89	**44.44**	50.00	14	0.126	13.39	0.0267
50.00	**55.56**	61.11	13	0.138	14.58	0.1917
61.11	**66.67**	72.22	9	0.110	11.64	0.7728
72.22	**77.78**	83.33	13	0.099	10.50	0.4805
83.33	**88.89**	94.44	6	0.033	3.50	0.8641
94.44	**100.00**	105.56	0	0.002087	0.221198	
Totals			106	0.991585	105.108	2.6895

Table A2.10: Calculation of Chi square for simulated wash performance.

The test statistic was then computed on the basis of a confidence level and the given number of bins, as shown in Table A2.11. A one sided test is used as we are only interested in finding whether or not the observed deviation from the model is larger than that due to chance.

DoF	6
Significance	80.00%
(Confidence)	20.00%
Test statistic	3.0701

Table A2.11: Calculation of test statistic.

The null hypothesis is that any differences between the simulated and observed data are purely chance, and this occurs if the measured Chi square is less than the test statistic. The alternative hypothesis is that there is a significant difference, i.e. that the simulation was a poor fit to the observed data.

In this case it is concluded that there is no significant difference between the simulation and observed data at the 80% significance level[188] using the Chi square test.

[188]The higher the significance the better the fit.

Appendix 3

Published papers

The following two papers had been accepted for publication at the time of submission of this thesis. They both include explanations of the Design for System Integrity (DSI) methodology.

A3.1 **IMechE paper**

RAINE J, PONS D, WHYBREW K

2001

The design process and a methodology for system integrity

Proceedings of the Institution of Mechanical Engineers Vol 215 part B, p569-576.

This paper has been published.

The design process and a methodology for system integrity

J K Raine[1], **D Pons**[2], and **K Whybrew**[3]
[1]Professor in Mechanical Engineering, University of Canterbury
[2]Head of Mechanical Engineering, Christchurch Polytechnic
[3]Associate Professor in Mechanical Engineering, University of Canterbury
Christchurch, New Zealand

Abstract: Common models of the design process tend not to reflect concurrent engineering, the influence of CAD on the design process, or the recursive divergent-convergent thinking processes at different levels of detail from whole system to sub-system to component design. Nor do these models illustrate the process of establishing functional integrity in the design of multi-component systems where large-scale interdependencies are present. This paper outlines design model developments by the authors and introduces a methodology for designing for system integrity and minimising risk from early in the conceptual phase, where uncertainty is high. The process involves probabilistic reasoning in propagating both qualitative and quantitative effects of changes or uncertainty in component or sub-system specifications through system models, and determining the integrity of the design from multiple viewpoints.

Keywords: Design process, design methodology, probabilistic reasoning, risk assessment, system integrity

1. INTRODUCTION

Research by the authors into the management of design and new product development in New Zealand, funded in part by New Zealand's Foundation for Research Science and Technology, is revealing [1] that few engineering designers in industry follow systematic design processes such as proposed, for example, by Pugh [2] and Pahl & Beitz [3]. This research [4,5] has also shown that communication is often lacking between product development team members working on different sub-systems. Together with an often predominantly intuitive design

process, this results in recurring cycles of task clarification, conceptual, embodiment and detail design, often quite late in a project. It is the authors' view [6] that a more systematic approach can reduce the incidence of late redesign work without adversely affecting the creative input to the design.

Diagrammatic design models, such as those of Pugh [2] and Pahl & Beitz [3], represent the sequence of events well for a simple design task, but would benefit from inclusion of more emphasis on customer needs, concurrent engineering, a systems approach to design and the integrative effect of CAD on conceptual, embodiment and detail design. (Studies by Dallas et al [4,5] show that many activities previously considered as detail design have been subsumed into embodiment design through the use of parametric solid modelling.)

Pugh's "Total Design" model [2] illustrates well the influence of factors in the design environment ("Elements of Specification"). Hubka and Eder [7, 8] give a comprehensive view of design science but their work is less accessible for practising engineering designers. They categorise modes of activity in design and present diagrammatic representations of activities and information flows in the design process. Crisp [9] represents the product development process as a set of parallel and iterative activities in a useful diagram called a "design reticule".

In the past 10 - 15 years there has been an increased focus on the product development process in the wider commercial context and on rapid product development [10,11,12]. Hales [13] illustrates the influence of the business environment and customer needs in his "Design in Context" diagram. Raine [14] has also presented a model for product innovation in the commercial context.

This paper presents a development of Pugh's [2] model that better illustrates the concurrent engineering environment, and then outlines developments by the authors of a methodology for ensuring system integrity from early in the conceptual phase of design. By system integrity is meant the full scope of cost and functional requirements, including for example performance, energy consumption, reliability, robustness (design where the optimum is a plateau tolerant of minor perturbations in design variables), and particular customer requirements.

2. PROCESS MODEL DEVELOPMENTS

Any development of existing design process models should recognise:

- That the product/plant/system design process is driven by customer requirements (and ongoing market research) throughout the design-manufacture-acceptance test period.
- The divergent/convergent thinking process that recurs during exploration of design alternatives (divergent), and evaluation and selection of a chosen design (convergent), at conceptual, embodiment and detail levels.
- The integrative effect of CAD on concept, embodiment and detail design, particularly where parametric solid modelling is used [4,5]. The latter now also routinely links into manufacture through production of CNC machine code from the solid model. In the modern multi-disciplinary product development team there are parallel and linking activities present between design, manufacturing planning, and customer-related activities (eg marketing) throughout the design process.

Fig. 1 illustrates a development of Pugh's [2] Total Design model that addresses some of these issues. The design requirement specification is determined from the defined market need. The design process then proceeds through the concept, embodiment and detail phases, and on to manufacture. In distinction to Pugh's earlier diagram, design and the planning of manufacture and marketing are shown proceeding concurrently, with communications

occurring between these areas of the company's operations. In the lower part of the diagram manufacturing moves to the forefront, and the pre-production and beta testing phases are shown preceding the market release. The diagram emphasises continual monitoring of the response of the design to market need, and the market is given prominence not shown by Pugh. The elements of specification represented by the outer circles are as in Pugh's original diagram. These are system integrity viewpoints that the designer must consider such as requirements of reliability, life, appearance, maintainability, working environment, etc.

The design process that occurs within each of the concept-embodiment-detail phases recurs at each level of increasing detail. This recursive process is addressed again in the next section.

[INSERT FIG. 1 HERE]

3. DESIGN FOR SYSTEM INTEGRITY (DSI)

Complex modern engineering systems, created in a multi-disciplinary rapid product development/concurrent engineering environment, demand a far higher level of systematic working and communications than in the past.

Knowledge-based software packages to assist the designer in exploring conceptual alternatives or in the analysis and optimisation of a given design are now common, for example in software models of complete motor vehicles. However, such methods have generally only been successful in specific domains, as they require relatively complete problem definition [14]. This is a characteristic not often available in the early design stages where uncertainty is high.

This section introduces a general methodology, for use by engineering designers or design managers, which ensures that system integrity is (a) being achieved, and (b) being maintained, as conceptual alternatives are explored early in the design process, and as design changes take place during concept, embodiment and detail phases.

In this problem domain, artificial intelligence methods have generally only been successful in well-defined applications that have fixed design strategies [16]. Bartsch-Sporl and Bakhtari [17] set out the requirements of artificial intelligence implementations as:

- Completeness of knowledge in the domain model
- Consistent logic within the domain
- Closed domain, not vulnerable to outside effects
- Problems that can be decomposed and recomposed
- Solution spaces that can be formally defined.

They point out that real life design domains seldom meet these requirements precisely, and their strategy has been to use artificial intelligence to assist the human designer rather than to solve design problems on its own.

One of the more successful implementations has been the "Schemebuilder" model of Bracewell et al [18], which helps the designer assess a design scheme from a functional viewpoint, principally by functional simulation. Rather than automating design using the expert system approach, the underlying philosophy is to provide decision support and allow the human designer to apply judgement. The system aims to provide the means to quickly evaluate alternative design solutions.

The approach developed in the present work also relies on the human designer for the creative input to the design, but establishes a Design for System Integrity (DSI) methodology to assist the human designer in decision-making and the management of design.

As an introduction to this process, Fig.2 illustrates the recursive activity that occurs at different levels of detail during the embodiment and detail design phases.Each sub-system or device in Fig. 2 will have properties of form, function, manufacturing method,cost; etc The total system is visualised as a 3-dimensional matrix of devices between which there will be relationships. These relationships could define exchanges (eg material, energy and information), or other more diffuse cause-consequence effects.. Iteration of the sub-system selection process is normally required, and once functional integrity has been established at the sub-system level the process in Fig. 2 windows down to the detail level where it recurs for the components within each sub-system.

[INSERT FIG. 2 HERE]

As the design progresses from concept to embodiment and detail levels, the focus on detail form and design for manufacture becomes greater. At the detail component level, geometry, manufacturing processes, tolerances and cost dominate the interactions with adjacent components. Internal to a component are relationships between different features that must be manufactured.

As the design is refined, sub-system concepts, embodiments and details are altered to an improved and more final state. At each level, the designer needs to understand how the relationships between sub-systems or components affect the overall integration of the design, and ultimately the cost, performance, robustness, reliability and other important characteristics of the system. Many of these relationships are intrinsically uncertain or qualitative. During this process, the combined effects of uncertainties in the values of parameters, or of a change in one part of the system on all other parts, should be evaluated, using analytical or qualitative design rules where possible. The DSI methodology outlined below has been developed to assist the designer in these tasks.

Fig. 2 illustrates just a single interaction, which is evaluated during DSI activity. In a multi-component system, a very large number of such interactions must be evaluated. The nature of these relationships, and the way in which they combine to cause an overall effect, is different at different levels of detail in the design process. A similar pattern of thinking is involved in Failure Mode and Effects Analysis (FMEA) and fault tree analysis, both of which are used to assist design managers [1,19].

The DSI methodology under development in the doctoral research of Pons is being configured as a computer software package that evaluates the integrity of a given engineering system. The central example used for illustration in the project is the specific case of an automatic dishwasher, for which integrity may be evaluated from multiple viewpoints such as wash performance, energy consumption, noise, ease of manufacture, cost, safety or reliability. A key feature of the methodology is its ability to propagate (at concept, embodiment or detail design stage) both qualitative and quantitative knowledge about design parameters through the DSI model. Fig. 3 gives an overview of the process.

[INSERT FIG. 3 HERE]

The DSI process is initiated by an expert, probably the designer, who identifies the key devices in the model. In the case of the dishwasher the DSI process system is defined typically with physical devices, eg “wash pump”, “sprayer”, etc. Each of these is attributed properties that relate to one or more of the viewpoints from which system integrity is to be evaluated, such as “wash pump cost”, “wash pump pressure”, and relationships are defined

for each of the multiple system integrity viewpoints. The designer then establishes the relationships between the devices. These relationships are different for each of the viewpoints under scrutiny, and can be either quantitative (numerical) or qualitative (textual).

Quantitative links give rise to numerical relationships, and are defined in terms of basic mathematical operators (addition, subtraction, multiplication, and division), or more complex mathematical functions (eg normal probability distributions or reliability functions such as the Weibull distribution) or even by logical functions (maximum, minimum).

Qualitative relationships are defined in textual terms. These could be pseudo-scalar values such as a temperature scale ("*hot, warm, cool, cold*") which has some linearity, or more abstract terms such as the type of soil present on dishware ("*baked-on, greasy, mashed potato, mixed*") in which no scale of precedence exists. In this work, a mapping process similar to that used in decision theory describes textual relationships. The expert defines this map at initial configuration of the system. For example, if wash performance ("*bad, marginal, acceptable, good, excellent*") of the dishwasher is dependent on *water temperature* and *soil type*, in a way which cannot be quantified, then the expert sets up a decision table which gives the resulting *wash performance* for every combination of the inputs *water temperature* and *soil type*. However, the expert does not give a single point deterministic estimate of the result, but gives the probability of each of the outputs. A specific example might be the combination of "*warm" water temperature* and "*greasy" soil type*. The expert gives a probability spread such *as 5% bad, 15% marginal, 35% acceptable, 20% good, 5% excellent*. This probability spread reflects the expert's uncertainty as to the precise relationship. In a similar way the expert provides a probability distribution for each of the other combinations.

This probabilistic decision theory approach means that any input parameters can be accommodated, including the textual descriptions used in the example. The need to assume any linearity or ranking of the inputs is avoided, and there is no need to apply weights or scores, as required, for example, in processes such as QFD. In the case of quantitative relationships (e.g. mathematical operators), there is no uncertainty, and a deterministic output may be described for, say, the addition of two numbers. The mapping method also permits any combination of textual and numerical relationship to be defined, and it is therefore possible to include both numerical and textual relationships in the same model.

Once the system has been defined by appropriate quantitative and qualitative relationships, the end user (who may be different from the expert), determines the input attributes. For example, the user might assert that *water temperature* is *warm*. This can then be propagated through the system to determine the resulting *wash performance* and other outputs. A more realistic assertion at early design stages might be to give a probability distribution e.g. *30% hot, 50% warm, 20% cool, 0% cold*, to encompass the uncertainty that exists at this stage. That is, the designer is leaning towards using a hot-warm set point on the thermostat, but wants to include the smaller possibility of a cool running option.

Once distributions are specified for all the inputs, the system outputs are computed as probability distributions for each parameter. If necessary, alarms (acceptance limits) can be defined by the user for any parameter, and these can be checked during computation. The end result is a set of distributions for each of the viewpoints defined.

The ability to propagate both qualitative and quantitative probabilistic reasoning, and process variability, through the system model means that

1 the DSI methodology can be applied to a number of different domains, and in particular, may be used in the early design stages when data are typically sparse.
2 system integrity may be assessed in a way that is impossible using conventional deterministic methods.

Moreover, the ability of the DSI methodology to propagate the effects of a change in a design parameter means that the "risk" in the result (e.g. total product cost) may be identified, quantifying how good or bad the outcome may be. Effects from various viewpoints may also be studied concurrently. For example, in the dishwasher, selection of a different circulation pump to reduce cost may alter water pressure and trigger responses from component properties relating to seal performance, system energy consumption, and water flow rates.

The computational methods will be described in detail in a forthcoming paper. Briefly, this novel process involves taking two input probability distributions, applying a quantitative operator (such as +, -, x, /, maximum, minimum) or qualitative map (using decision theory) to determine an output distribution. The software engine applies probabilistic reasoning in a combinatorial fashion.

This is a departure from established techniques, where the effect of variability in multiple parameters affecting an overall outcome, and the associated numerical relationships, have commonly been handled using Monte Carlo methods [20,21]. However Monte Carlo methods are unable to handle qualitative inputs. The DSI numerical method avoids Monte Carlo simulation altogether, although it has similar functionality and computational power. It also provides the functionality of decision theory.

An example of DSI is the determination of Life Cycle Cost of a dishwasher development. The Producer Profit (in terms of present value) is the net profit less the Venture Costs. Each of these may be further broken down to take into account product development cost, plant commissioning and de-commissioning, overheads, production cost, warranty failures, liability costs, and sales volume, though these are not illustrated here. Each of these has a probability distribution of its own, and these are added together. The summation process at the lower levels is not shown in this illustration. The final step in the analysis (shown in Fig. 4) is to subtract the probability distribution for *Venture Costs* from that for *Net Total Profit*, to obtain *the Producer Profit*. The charts represent probability densities.

[INSERT FIG. 4 HERE]

In the early stages of design, the probability distributions associated with key quantitative parameters will tend to be flatter and will propagate more uncertainty through the model. In addition, the functional relationships themselves are more uncertain. Uncertainty reduces as the design progresses to more concrete stages and more prototype test results become available. This results in greater certainty about functional relationships and less process variability, both of which result in narrower distributions of final values, and a corresponding clarity of risk and design integrity. However uncertainty may remain if process variability remains significant, although this is more the case with numerical inputs than in the textual example of *water temperature* given earlier.

The combined effect of the probability distributions on values of a large number of parameters can be determined to show the overall robustness of the design. By inference, the DSI methodology is able to identify sensitivity of the system design to particular parameters and how tightly tolerances in any given parameter must be held to achieve a robust design.

As the DSI methodology produces a probability distribution for each parameter in the model, this distribution may be used in project management, particularly to make decisions regarding viability and risk. Unlike expert system tools where the software seeks to make a decision using artificial intelligence method, the DSI method does not make decisions itself. The method is an analytical assessment tool which supports the human decision making process by providing information about risk probability. Unless numerical risk acceptance

criteria are incorporated in the model, it is up to the human manager to make the decisions, based on his/her own tolerance of or aversion to risk.

4. CONCLUSIONS

While common diagrammatic models of the design process are sound, there is a need for these to be updated to emphasise customer needs and concurrent engineering, which are the norm in rapid product development environments.

Designers can have difficulty simultaneously grasping the multiple relationships between systems, sub-systems and components. This paper has outlined a Design for System Integrity (DSI) methodology, which is under development as a software tool to assist in faster evaluation of system integrity from multiple viewpoints, even at early conceptual stages. The methodology fuses quantitative and qualitative assessment, and incorporates probabilistic reasoning. Implementation of the DSI method, in its quantitative and qualitative aspects and its multi-viewpoint capability, will be illustrated in more detail in forthcoming papers.

5. REFERENCES

1. **Shaw A.**, **Aitchison, D.R., Raine, J.K., Whybrew, K.** Summary of results of the survey, "Rapid Product Development for World Class Manufacturing". Technical Report No. 58, Department of Mechanical Engineering, University of Canterbury, May 2000, 23pp.
2. **Pugh, S**. *Total Design, Integrated Methods for Successful Product Engineering*, Addison Wesley, 1991.
3. **Pahl, G.** and **Beitz W.** *Engineering Design*, London: The Design Council, and Berlin/Heidelberg, Springer-Verlag, 1984.
4. **Dallas, T.P.** Rapid development of high quality products, M.E. Thesis, Department of Mechanical Engineering, University of Canterbury, January 1999
5. **Whybrew K., Raine J.K**. and **Dallas T.P.** The Application of Design Phase Diagrams in the Management of Design of Telecommunications Products, *Proceedings of the 12th International Conference on Engineering Design*, Munich 24-26 August 1999, **3**, 1519-1524.
6. **Raine J.K.** and Whybew K. Speeding up product development through systematic management of design. *World Manufacturing Congress, Proceedings of the International Symposium on Manufacturing Management.* Auckland, November 1997, ICSC Press Canada, pp98-105
7. **Hubka, V.** and **Eder W.E.** *Theory of Technical Systems*, Springer-Verlag, 1988.
8. **Hubka, V.** and **Eder W.E.** *Design Science*, Springer-Verlag, 1996.
9. **Crisp, J.D.C.** Industrial innovation and Engineering Education Part III - Acquiring Sensitivity to Innovation, *Mechanical Engineering Transactions of the IEAust*., 1986, 454-49.
10. **Ulrich K.T.** and **Eppinger S.D.** *Product Design and Development*, McGraw Hill, 1995.
11. **Smith P.G**. and **Reinertsen D.G.** *Developing Products in Half the Time*, Van Nostrand Reinhold, 1991.
12. **Cooper R.G**. *Winning at New Products - Accelerating the Process from Idea to Launch*, 2nd Edition, Addison Wesley, 1993.
13. **Hales, C.** *Managing Engineering Design*, Longman Scientific & Technical, 1993.
14. **Raine, J.K.** A Focus on Mechanical Engineering Design at the University of Canterbury. *Journal of Engineering Design*, **5** (4), 1994, 353-366.

15. **Candy, L., Edmonds, E.** and **Patrick, D**. Interactive knowledge support to conceptual design, in Sharpe J. (ed), Artificial intelligence system support for concept design. *Proceedings of the 1995 Lancaster International Workshop on Engineering Design*, Springer, 1996, 260-278.
16. **Tang M.** Development of an integrated AI system for conceptual design support, in Sharpe. J (ed), Artificial intelligence system support for conceptual design. *Proceedings of the 1995 Lancaster International Workshop on Engineering Design*, Springer, 1996, 153-169.
17. **Bartsch-Sporl, B.** and **Bakhtari, S.** A support system for building design - experiences and convictions from the FABEL project, in Sharpe J (ed), Artificial intelligence system support for conceptual design. *Proceedings of the 1995 Lancaster International Workshop on Engineering Design*, Springer, 1996, 279-297.
18. **Bracewell, R.H., Chaplin R.V., Langdon P.M., Li M., Oh V.K., Sharpe J.E.E.,** and **Yan X.T**. Integrated platform for AI support of complex design (Part 2): Supporting the embodiment process, in Sharpe J. (ed), Artificial intelligence system support for conceptual design. *Proceedings of the 1995 Lancaster International Workshop on Engineering Design*, Springer, 1996,189-207.
19. **Araujo, C.S., Benedetto-Neto, H., Campello, A.C., Segre, F.M. and Wright, I.C.** The utilisation of product development methods: a survey of UK industry. *Journal of Engineering Design*, 1996, **7**(3), 265-277
20. **Kleijnen, J.P.C**. Verification and validation of simulation models, *European Journal of Operational Research*, Elsevier Science B.V. 1995, **82**(1), 145-162.
21. **Zhang, S., McAllister J.,** and **Dorricott, MG**. Probabilistic analysis of risk in mining projects - as easy as 123. *Proceedings of Third Large Open Pit Mining Conference, Australasian Inst of Mining & Metallurgy*, Aug 30-Sep 3 1992, 455-459.

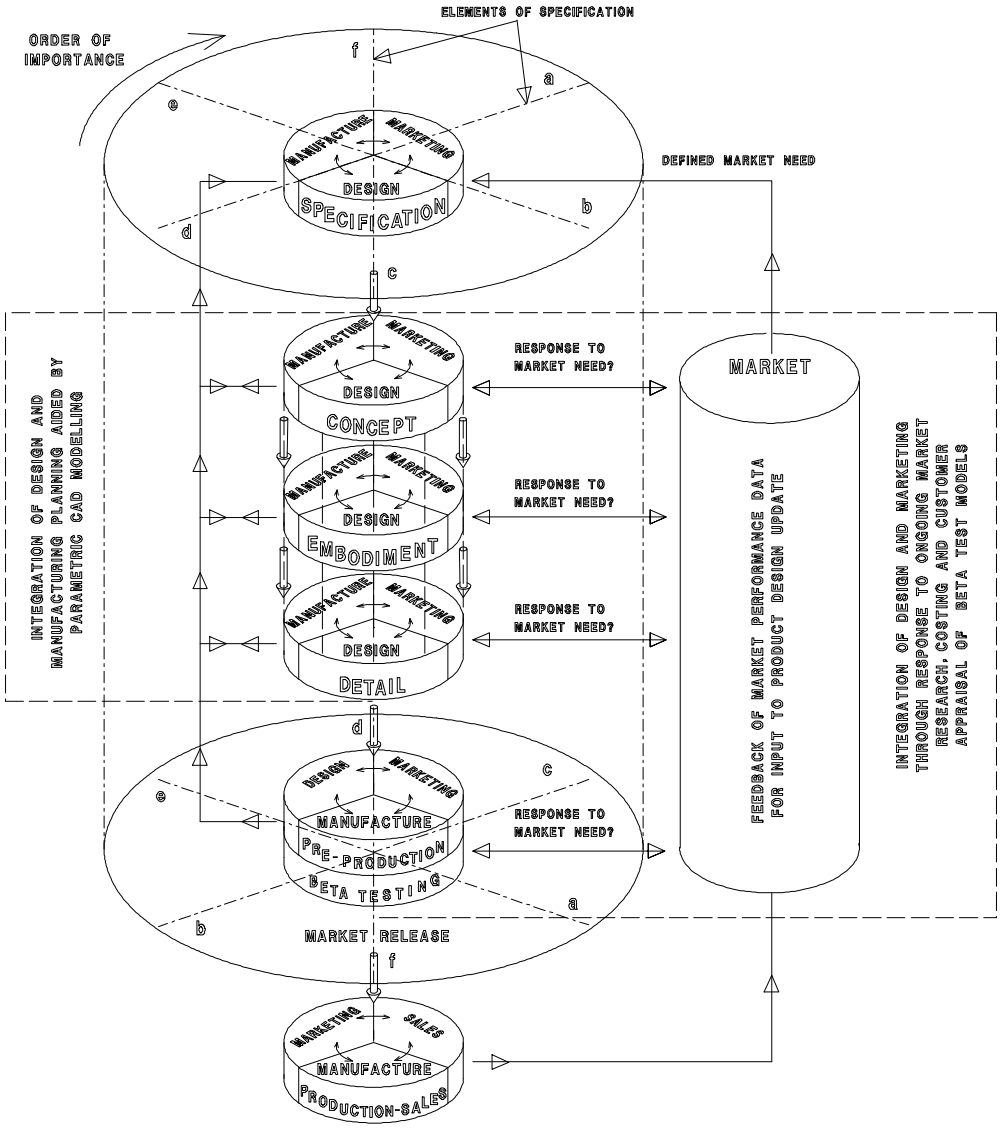


Fig. 1 Development of Pugh [1] Total Design activity model with concurrent design, manufacturing planning and market development, plus increased focus on customer needs and

the market

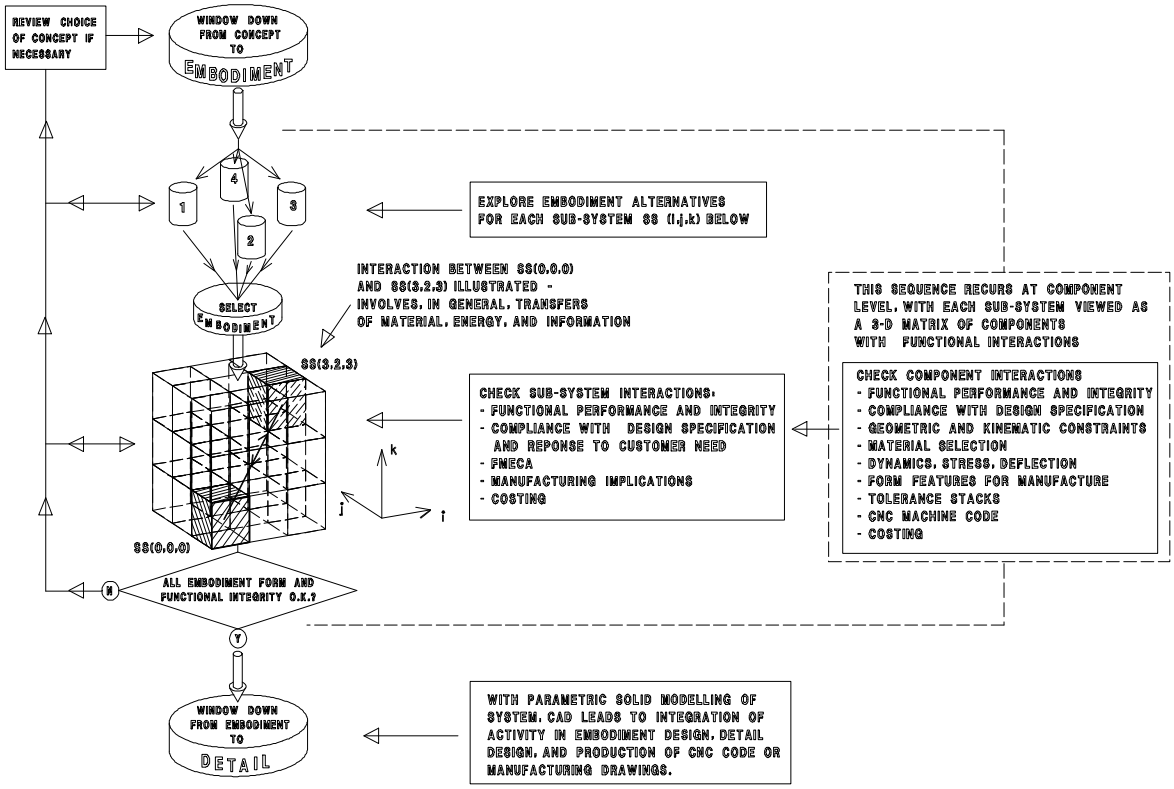


Fig. 2 Recursive process of exploring alternatives, plus sub-system and component interactions at embodiment and detail level

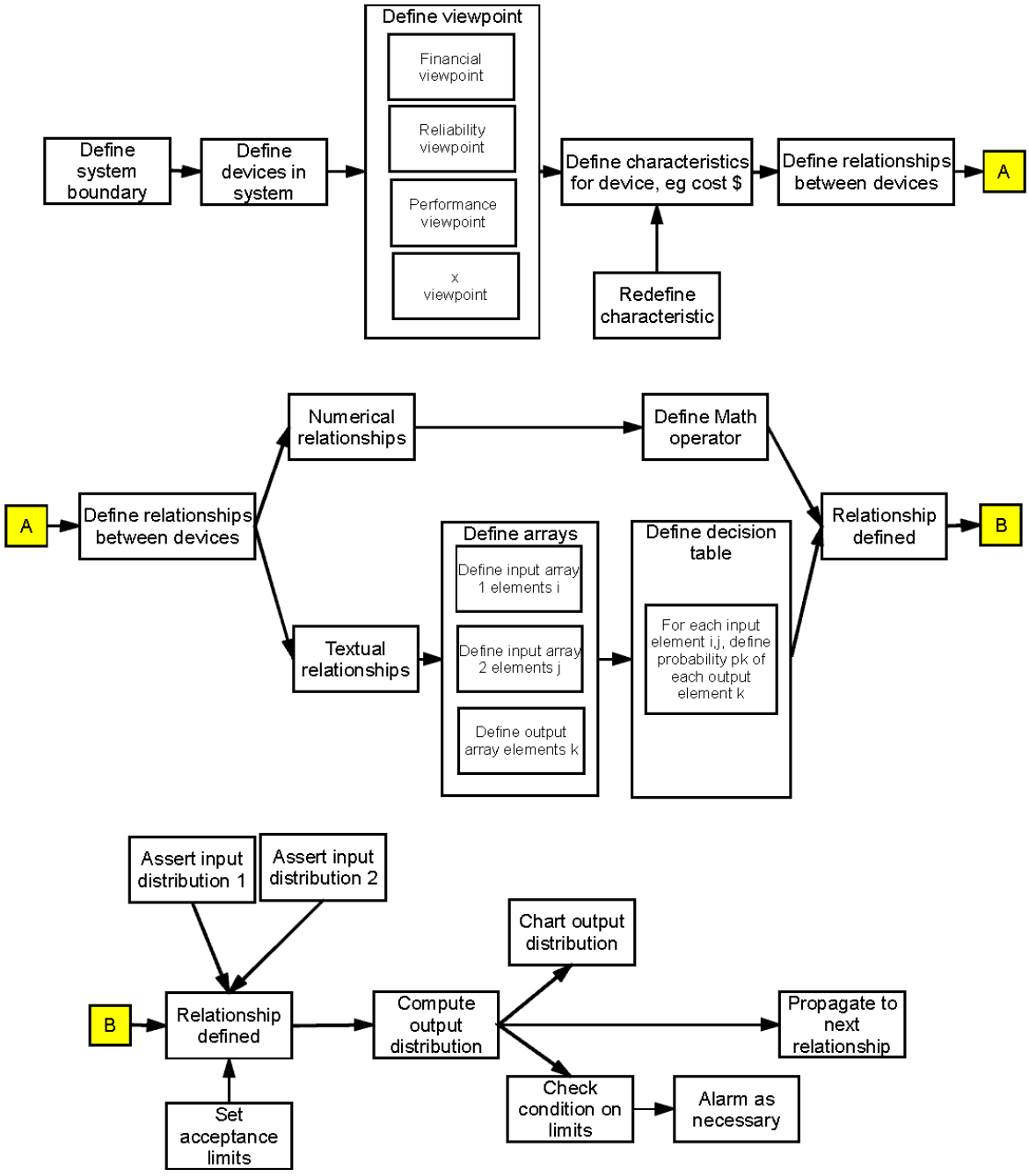


Fig. 3 Flow chart illustrating core process of the Design for System Integrity (DSI) method

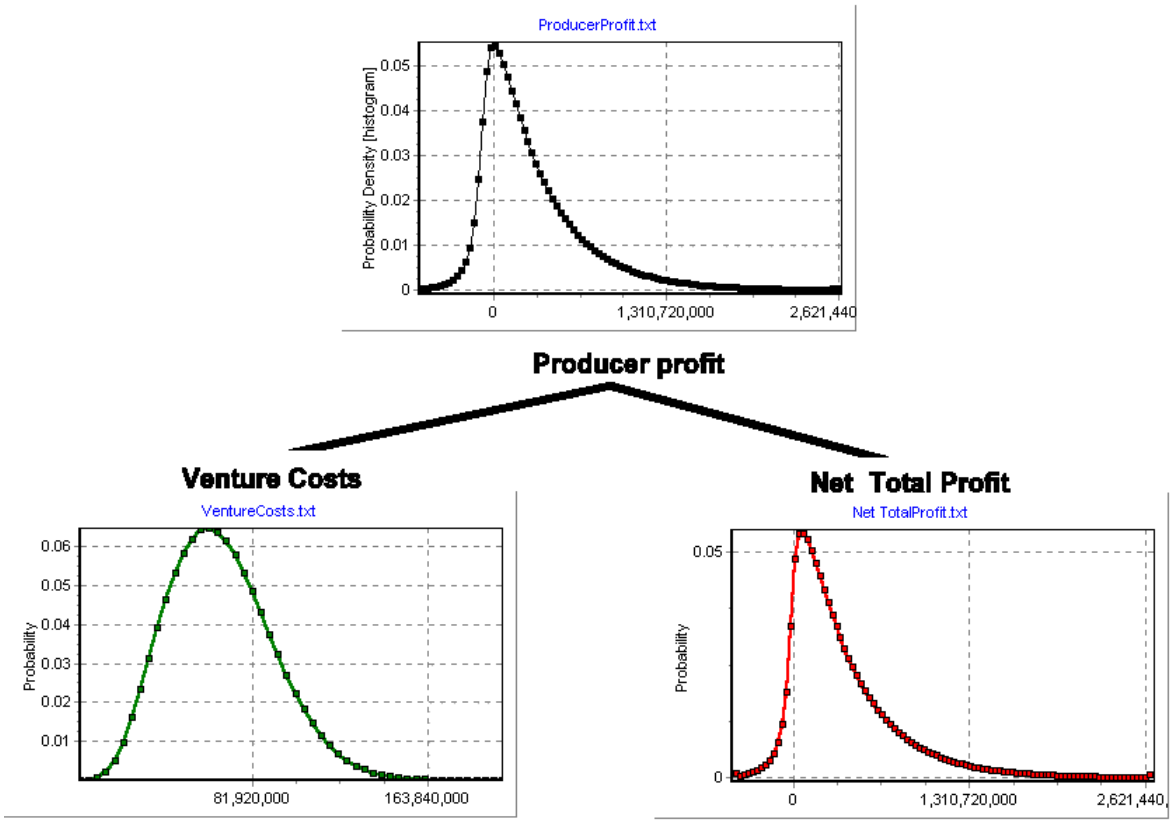


Fig. 4 Combination of product lifetime profit and venture costs for investment appraisal. The horizontal axes are cost (eg in dollars NPV) and the vertical axes represent probability density. The figures should therefore be interpreted as histograms, showing the distributions over which the results are likely to occur. In particular, note in the top chart the presence of a tail below zero dollars: this shows the risk of making a loss on this business proposition.

A3.2 **ICED paper**

The second paper is '**A METHODOLOGY FOR SYSTEM INTEGRITY IN DESIGN AND MANAGEMENT**' (John K. Raine, Dirk Pons, Ken Whybrew) and has passed peer review and will be presented at the *International Conference on Engineering Design ICED 01 Glasgow, August 21-23, 2001.*

INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN
ICED 01 GLASGOW, AUGUST 21-23, 2001

A METHODOLOGY FOR SYSTEM INTEGRITY IN DESIGN AND MANAGEMENT

John K. Raine, Dirk Pons, Ken Whybrew

Keywords: Design process, knowledge-based systems, probabilistic design, system integrity

1 Introduction

This paper introduces a methodology for designing for system integrity [1] in multi-component systems and minimising risk from early in the conceptual phase, where uncertainty is high. The process involves probabilistic reasoning in propagating both qualitative and quantitative effects of changes, or uncertainty in component or sub-system specifications, through system models, and determining the integrity of the design from multiple viewpoints. The methodology is very versatile and the software that has been developed is able to handle not only Design for System Integrity (DSI), but also a number of other tasks, such as evaluating system reliability, investment appraisal under uncertainty for engineering and other projects, and risk assessment in project management. The software, which has already been demonstrated for a number of these applications, is used as an aid to design management.

2 The Design Process and Knowledge-Based Systems in Design

A systems approach to design involves the identification of a functional model for the engineering system and the creation of a number of modules or sub-systems and components, between which there are defined interfaces and transfers of material, energy and information. As the design is refined, sub-system concepts, embodiments and details are altered to an improved and more final state. At each level, the designer needs to understand how the

relationships between sub-systems or components affect the overall integration of the design, and ultimately the cost, performance, robustness, reliability and other important characteristics of the system.

Research at the University of Canterbury into the management of design and new product development in New Zealand [2] indicates that few engineering designers in industry follow systematic design processes such as proposed, for example, by Pugh [3] and Pahl & Beitz [4]. This research [5] has also shown that communication is often lacking between product development team members working on different sub-systems. This results in recurring cycles of task clarification, conceptual, embodiment and detail design, often quite late in a project.

Raine et al [6] have proposed diagrams to better represent concurrent activity between customer-focused design, manufacturing and marketing functions in the design process, and to illustrate the recursive process of exploring sub-system and component interrelationships. These relationships, which will be numerous in a multi-component system, are examined from various viewpoints, such as in Pugh's Elements of Design Specification [3]. The objective of the research described in this paper was to develop a methodology and software to operate as an aid to design managers in determining the integrity of a design from multiple viewpoints, for example whole life cycle cost, performance, noise level, appearance, manufacturability, maintainability, or reliability. It was desired to be able to evaluate rapidly, using analytical or qualitative design rules, the effect of specification uncertainty or changes (design alternatives) in one component or sub-system on other components and on the whole system, from the different viewpoints, and, importantly, at the early conceptual design stage when there is a high level of uncertainty.

Antonsson et al [for example, reference 7] have presented notable work on methods for incorporating imprecision in engineering design decision-making. Antonsson's Method of Imprecision allows for uncertainty at early design stages using fuzzy calculus but does not appear to propagate qualitative effects through the design model. Pons [1] has compared the features of Antonsson's method with the DSI methodology described below.

The research of Pons [1] highlighted a need for design models or methodologies that can

- explore how the design behaves [8]
- handle non-numeric parameters and evaluation of tolerances
- evaluate a configuration without being forced to assign values to the attributes
- use a symbolic representation in preliminary design when embodiment and geometry may be uncertain [8]
- measure and quantify lifetime performance of designs.

Knowledge-based software packages to assist the designer in exploring conceptual alternatives or in the analysis and optimisation of a given design are now common, for example in software models of complete motor vehicles. However, such methods have generally only been successful in specific domains, as they require relatively complete problem definition [9], which is often unavailable in the early design stages where uncertainty is high.

In this problem domain, artificial intelligence methods have generally only been successful in well-defined applications that have fixed design strategies [10]. Bartsch-Sporl and Bakhtari [11] set out the requirements of artificial intelligence implementations as:

- Completeness of knowledge in the domain model
- Consistent logic within the domain
- Closed domain, not vulnerable to outside effects
- Problems that can be decomposed and recomposed
- Solution spaces that can be formally defined.

They point out that real life design domains seldom meet these requirements precisely, and their strategy has been to use artificial intelligence to assist the human designer rather than to solve design problems on its own.

An example of a more successful implementation has been the “Schemebuilder” model of Bracewell et al [12], which helps the designer assess a design scheme from a functional viewpoint, principally by functional simulation. Rather than automating design using the expert system approach, this system provides decision support and allows the human designer to apply judgement and quickly evaluate alternative design solutions.

3 Design for System Integrity Methodology

The DSI methodology and software belongs to the family of knowledge-based systems that help the designer assess a design scheme from a functional viewpoint. The DSI methodology propagates (at concept, embodiment or detail design stage) both quantitative and qualitative knowledge about design parameters through a tree structure not dissimilar to a fault tree or decision tree. The process is outlined in the flow diagram of Figure 1.

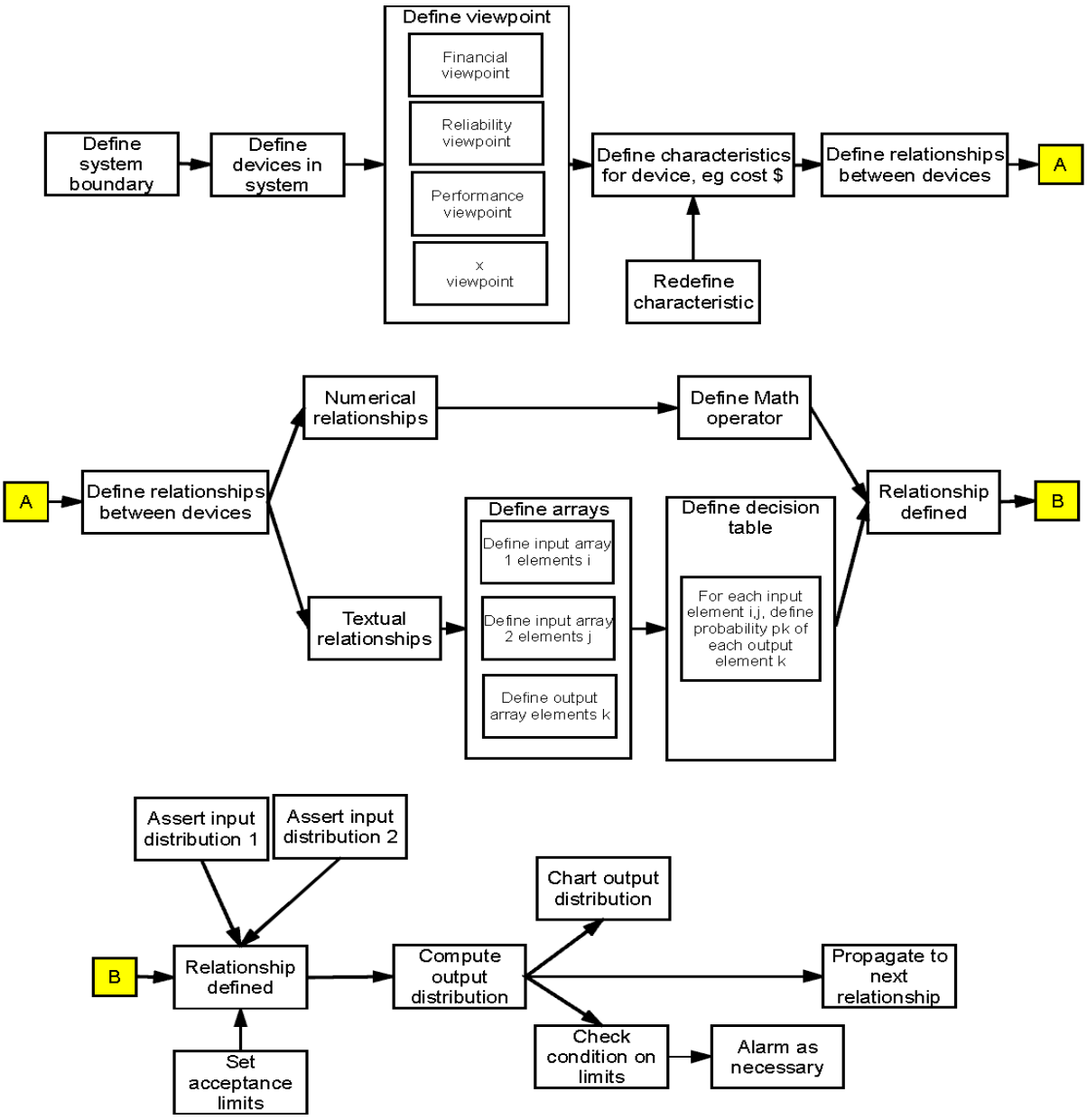


Figure 1 Flow chart illustrating core process ofthe Design for System Integrity (DSI) method

Space does not permit a comprehensive explanation and illustration of the methodology, which has been fully described by Pons [1], and which will be published elsewhere in more detail. However a brief example is presented below.

The research used as a key example the design of automatic dishwashers. The designer identifies key devices in the model. In the case of the dishwasher the DSI process system is defined typically with physical devices, eg *"wash pump"*, *"sprayer"*, etc. Each of these is attributed properties that relate to one or more of the viewpoints from which system integrity is to be evaluated, such as *"wash pump cost"*, *"wash pump pressure"*. The designer then establishes the relationships between the devices. These relationships are different for each of

the viewpoints under scrutiny, and can be either quantitative (numerical, e.g. fluid power = pressure x volume flow rate), or qualitative (textual). Wash performance of the dishwasher is modelled as 100% less the soil demerit points. The overall effectiveness of soil removal is represented by a substantial and complex DSI tree structure which has been fully described by Pons [1]. Space here only allows for brief examples of quantitative and qualitative DSI processes.

Quantitative links give rise to numerical relationships, and are defined in terms of basic mathematical operators (addition, subtraction, multiplication, and division), or more complex mathematical functions (eg normal probability distributions or reliability functions such as the Weibull distribution) or even by logical functions (maximum, minimum). The elemental process in the DSI methodology involves taking two discretised input probability distributions, applying a quantitative operator (such as +, -, x, /, maximum, minimum) or qualitative map (using decision theory) to determine an output distribution.

In the wash performance model one of the elemental quantitative combinations in the sub-system computation for soil particle demerits evaluates the water retained in the machine at the end of the wash cycle as the sum of the water retained in the sump and the water retained as a film on the wash cavity. The fragment of the DSI tree containing this operation is shown in Figure 2.

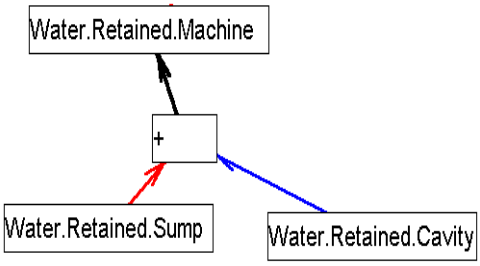


Figure 2: Fragment of DSI computation for soil particle demerits within wash performance tree.

The inputs in this case are represented respectively by normal and Weibull probability distributions. The addition of the two discretised probability distributions, represented as histograms, is illustrated in Figure 3. The process can be computationally demanding. In this example, if the two probability distributions were each approximated by 100 discrete values, there will be 10000 calculations to form the combined distribution, and there may be many such combinations in the overall DSI analysis tree just for the wash performance. In the retained water computation each of the 10,000 additions has an associated joint probability being the product of the probabilities of the component water volumes. The overall joint probability for a specific value of overall retained water volume is the sum of the various joint probabilities associated with the occurrences of that specific retained water volume as an outcome.

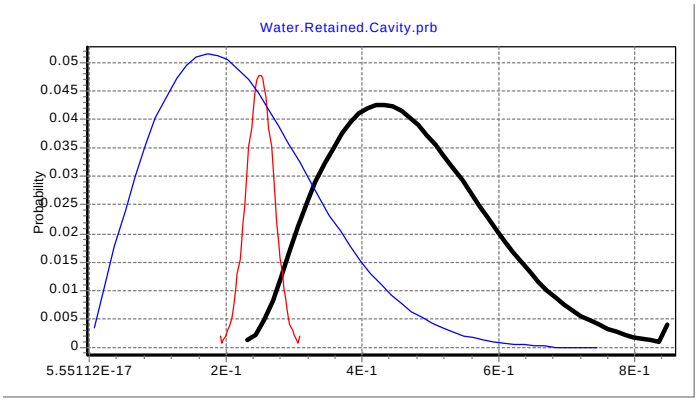


Figure 3: Sum of sump (normal) and cavity (Weibull) retained water probability distributions to give an output as the bold line. The horizontal axis is volume in litres and the vertical scale probability density

Qualitative parameters are defined in textual terms. These could be pseudo-scalar values such as a temperature scale ("*hot, warm, cool, cold*"), or more abstract terms such as the type of soil present on dishware (e.g."*sauce, mashed potato, breakfast cereal*") and soil thermal treatment ("*fresh, dried, reheated, baked, burned"*), where no scale of precedence exists. A mapping process similar to that used in decision tables describes textual relationships and enables qualitative information to be propagated through the DSI tree. The expert defines this map at initial configuration of the system.

For example, if wash performance ("*bad, marginal, acceptable, good, excellent*") of the dishwasher is dependent on *soil thermal treatment* and *soil type*, in a way which cannot be quantified, then the expert sets up a decision table which gives the resulting wash performance for every combination of the inputs *soil thermal treatment* and *soil type*. However, the expert does not give a single point deterministic estimate of the result, but gives the probability of each of the outputs. A fragment of a wash performance map relationship is shown in Figure 4, in which the map takes *soil thermal treatment* and *soil type*, and combines them to produce the output. A *soil thermal treatment* model is shown in Figure 5. In the example shown, all the soil is taken as *dried*, which therefore has a probability of 1. A *soil type* profile is shown in Figure 6, where the proportions have been determined from the relative masses of the soils.

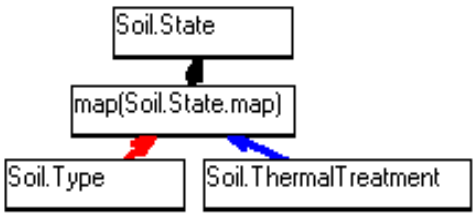


Figure 4: Representation of a map Relationship in the Design Integrity Software

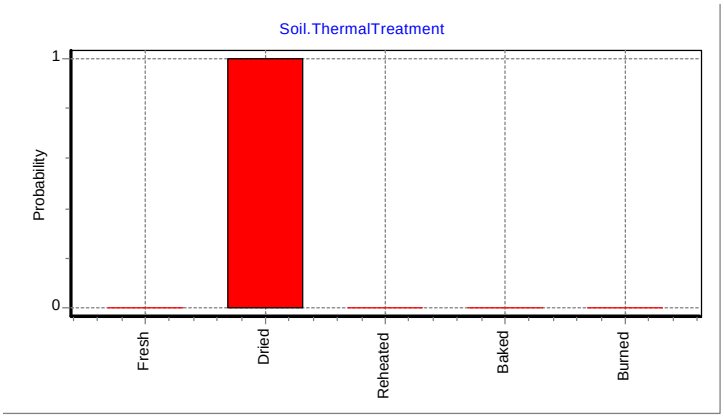


Figure 5: The profile for *soil thermal treatment* can range from “fresh” to “burned” soil.

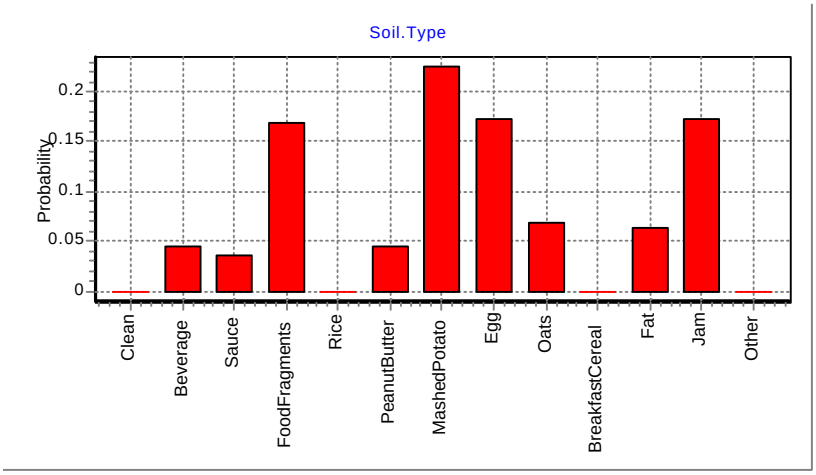


Figure 6: *Soil type* shown as a histogram based on mass of various soils used in standard dishwasher performance tests.

n the example illustrated here, *soil state* is computed by applying a subjective map to *soil type* and *soil thermal treatment.* A fragment of this map is shown in Figure 7. A specific example within Figure 7 is that in the case of egg, the likelihood would be 50% as soluble film and 50% as sticky paste. The output of the map process is s*oil state* as shown in Figure 8.

Soil.State			Soil.Type												
			Clean	Bever	Sauce	FoodF	Rice	Peanut	Mashe	Egg	Oats	Break	Fat	Jam	Other
Soil.Therm	Fresh	Comment													
		Clean	1	0	0	0	0	0	0	0	0	0	0	0	0
		SolubleFilm	0	0.333	0.1	0	0	0	0	0.5	0	0	0	0.5	0.142
		FineParticulate	0	0.333	0.2	0.25	0	0	1	0	0.333	0.333	0	0	0.142
		LoosePieces	0	0	0.2	0.5	1	0	0	0	0.333	0.333	0	0	0.142
		StickyPaste	0	0	0.3	0.25	0	1	0	0.5	0.333	0.333	0	0.5	0.142
		Greasy	0	0.333	0.2	0	0	0	0	0	0	0	1	0	0.142
		DriedStuck	0	0	0	0	0	0	0	0	0	0	0	0	0.142
		CookedOn	0	0	0	0	0	0	0	0	0	0	0	0	0.142
		BurnedOn	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7: Map to convert *soil type* and *soil thermal treatment* into *soil state*. Numbers are probabilities where any one column will sum to unity.

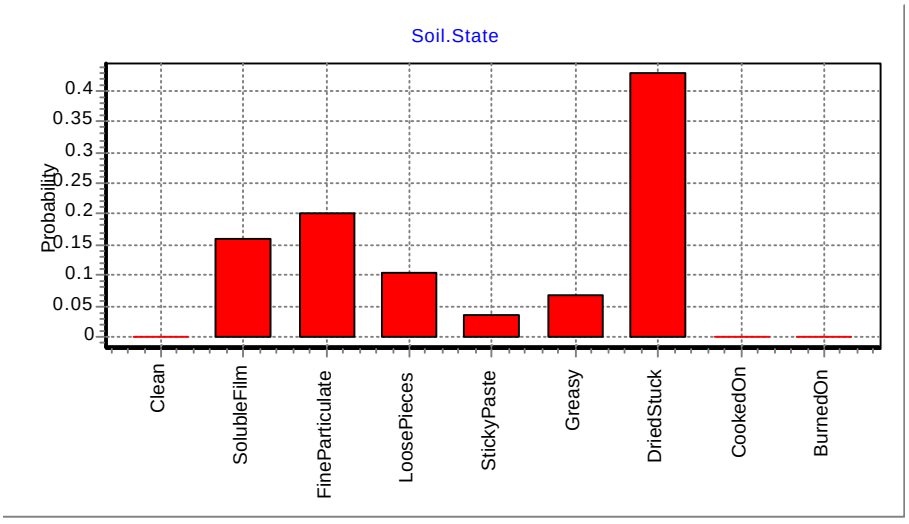


Figure 8: *Soil state* is a combination of *soil type* and *soil thermal Treatment*. It describes the important wash characteristics of the soil.

Once the system has been defined by appropriate quantitative and qualitative relationships, the end user (who may be different from the expert designer) determines the input attributes. For example, the user might assert that *water temperature* is *warm*. This can then be propagated through the system to determine the resulting *wash performance* and other outputs. A more realistic assertion at early design stages might be to give a probability distribution e.g. *30% hot, 50% warm, 20% cool, 0% cold*, to encompass the uncertainty that exists at this stage. That is, the designer is leaning towards using a hot-warm set point on the thermostat, but wants to include the smaller possibility of a cool running option.

After distributions are specified for all the inputs, the system outputs are computed as probability distributions for each parameter. If necessary, alarms (acceptance limits) can be

result is a set of distributions for each of the viewpoints defined,
or each viewpoint from which the designer wishes to evaluate the integrity of the system.

tree and the DSI software then computes an outcome probability distribution for the viewpoint concerned, e.g. overall wash performance, product cost or product reliability. Effects from

performance of the dishwasher, selection of a different circulation pump to reduce cost may alter water pressure and trigger responses from component properties relating to seal

The DSI software engine applies probabilistic reasoning in a combinatorial fashion and allows both numerical and textual relationships in the same model. This is a departure from established

and the associated numerical relationships, have commonly been handled using Monte Carlo methods [13]. The DSI numerical method avoids Monte Carlo simulation altogether, although

decision theory. The ability of the DSI methodology to propagate the effects of a change in a design parameter means that the "risk" in the result (e.g. total product cost) may be identified,

software seeks to make a decision using artificial intelligence method, the DSI method does not make decisions itself. The method is an analytical assessment tool that supports the human

The DSI software has been created using Delphi programming language. In the operating DSI software there are approximately 40,000 lines of code for a comprehensive package that can

4. Conclusions

systems, sub-systems and components. The Design for System Integrity (DSI) methodology has been developed as a software tool to assist in faster evaluation of system integrity from multiple

methodology fuses quantitative and qualitative assessment, and incorporates probabilistic reasoning.

variability, through the system model means that

- particular, may be used in the early conceptual design stages when data are typically sparse.
  System integrity may be assessed in a way that is impossible using conventional deterministic methods.

References

[1] Pons, D. A methodology for system integrity in design. PhD thesis, University of Canterbury, Christchurch, NZ, March 2001.

[2] Shaw A., Aitchison, D.R., Raine, J.K., Whybrew, K. Summary of results of the survey, "Rapid Product Development for World Class Manufacturing". Technical Report No. 58, Department of Mechanical Engineering, University of Canterbury, May 2000, 23pp.

[3] Pugh, S. *Total Design, Integrated Methods for Successful Product Engineering.* Addison Wesley, 1991.

[4] Pahl, G. and Beitz W. *Engineering Design.* London: The Design Council, and Berlin/Heidelberg, Springer-Verlag, 1984.

[5] Whybrew K., Raine J.K. and Dallas T.P. The Application of Design Phase Diagrams in the Management of Design of Telecommunications Products, *Proceedings of the 12th International Conference on Engineering Design,* Munich 24-26 August 1999, 3, 1519-1524.

[6] Raine, J.K., Pons, D., and Whybrew, K. The design process and a methodology for system integrity. *Short Communications in Manufacture and Design, Proc. IMechE Part B,* 2001 (to appear)

[7] Antonsson, E.K. & Otto, K.N. Imprecision in Engineering Design. *Journal of Mechanical Design, Transactions of the ASME,* 1995, **117B**, 25-32.

[8] Finger, S. and Dixon, J. A review of research in mechanical engineering design. Part 2: Representations, analysis and design for the life cycle. *Research in Engineering Design,* **1**, 121-137, 1989.

[9] Candy, L., Edmonds, E. and Patrick, D. Interactive knowledge support to conceptual design, in Sharpe J. (ed), Artificial intelligence system support for concept design. *Proceedings of the 1995 Lancaster International Workshop on Engineering Design,* Springer, 1996, 260-278.

[10] Tang M. Development of an integrated AI system for conceptual design support, in Sharpe. J (ed), Artificial intelligence system support for conceptual design. *Proceedings of the 1995 Lancaster International Workshop on Engineering Design,* Springer, 1996, 153-169.

[11] Bartsch-Sporl, B. and Bakhtari, S. A support system for building design - experiences and convictions from the FABEL project, in Sharpe J (ed), Artificial intelligence system support for conceptual design. *Proceedings of the 1995 Lancaster International Workshop on Engineering Design*, Springer, 1996, 279-297.

[12] Bracewell, R.H., Chaplin R.V., Langdon P.M., Li M., Oh V.K., Sharpe J.E.E., and Yan X.T. Integrated platform for AI support of complex design (Part 2): Supporting the embodiment process, in Sharpe J. (ed), Artificial intelligence system support for conceptual design. *Proceedings of the 1995 Lancaster International Workshop on Engineering Design*, Springer, 1996,189-207.

[13] Kleijnen, J.P.C. Verification and validation of simulation models, *European Journal of Operational Research*, Elsevier Science B.V. 1995, **82**(1), 145-162.

References

ACKOFF RL
1962
Scientific method optimising applied research decisions
John Wiley & Sons, New York.

ANDERSSON K.
1994
Vocabulary for conceptual design
Proceedings of the IFIP TC5/WG5.2 Workshop on Formal Design Methods for CAD
IFIP Transactions B: Computer Applications in Technology B18 p 157- 171

ADDY EA, SIMMS LJ
1996
Experience Report: The Use of Functional Flows in IV&V
http://research.ivv.nasa.gov/~eaddy/funcflow

ALTENBACH TJ
1995
Comparison of risk assessment techniques from qualitative to quantitative
American Society of Mechanical Engineers, Pressure Vessels and Piping Division (Publication) PVP v 296 p 15-28

ANDERT E, PETERS L
1994
Computer-aided system design assessment tool
Proceedings of the 1994 IEEE 13th Annual International Phoenix Conference and Communications Phoenix, AZ, USA, p 193 - 199

ANSELL J, AL-DOORI M
1993
ARDA - an expert system for reliability data analysis
Corporate Source: Univ of Edinburgh, Edinburgh, UK Conference Title: Proceedings of the International Conference on APL, APL Quote Quad v 24 n 1 Aug p 1-5

ANSI/AHAM DW-1-1992
1992
Household electric dishwashers
American National Standard, Association of Home Appliance Manufacturers, 20 North Wacker Drive, Chicago, Illinois 60606

ANTONSSON EK, OTTO KN
1995
Imprecision in engineering design
Journal of Mechanical Design, Transactions Of the ASME 117B Jun p 25-32

ASHLEY S
1993
Failure analysis beats Murphy's law
Mechanical Engineering v 115 n p 70-72

BACH J
1999
Risk and requirements-based testing
Computer v 32 n 6 1999 IEEE p 113-114

BANARES ALCANTARA R, KING J, BALLINGER G
1996
EGIDE: A design support system for conceptual chemical process design.
In SHARPE J (ed), A I system support for conceptual design. Proceedings of the 1995 Lancaster International Workshop on Engineering Design, Springer, p 139 - 152

BARTSCH-SPORL B, BAKHTARI S
1996
A support system for building design - experiences and convictions from the FABEL project.
In SHARPE J (ed), A I system support for conceptual design. Proceedings of the 1995 Lancaster International Workshop on Engineering Design, Springer, p 279-297

BASU A
1998
Practical risk analysis in scheduling
AACE International. Transactions of the Annual Meeting Jun 28-Jul 1 1998 4p

BAZU M
1995
Combined fuzzy-logic & physics-of-failure approach to reliability prediction
IEEE Transactions on Reliability 44 2 p 237-242

BELEV GC
1992
Design management--Begin at the beginning
Proceedings of the Annual Reliability and Maintainability Symposium Mar, Publ by IEEE, p 98-100

BERGMAN B, KLEFSJÖ B
1994
Quality: from customer needs to customer satisfaction
Studentlitteratur, Sweden, ISBN 91-44-46331-6

BERLEANT D, KUIPERS BJ
1997
Qualitative and quantitative simulation: bridging the gap
Artificial Intelligence v 95 n 2 Sep p 215-255

BERNHARDT A, WOLVERTON MP
1996
Managing risk in new product development
Technical Papers, Regional Technical Conference - Society of Plastics Engineers Sep 25-26 1996 p C1-C5

BIONDO SJ
1990
Fundamentals of expert systems technology
Ablex Publishing, New Jersey

BLANCHARD BS, FABRYCKY WJ
1998
Systems engineering and analysis
Third edition, Prentice-Hall Inc, USA, ISBN 0-13-135047-1

BOX G, BISGAARD S, FUNG C
1988
An explanation and critique of Taguchi's contributions to quality engineering.
Report No. 28, Center for Quality and Productivity Improvement, University of Wisconsin-Madison, 610 Walnut ST, Madison WI 53705 USA

BRACEWELL RH, CHAPLIN RV, LANGDON PM, LI M, OH VK, SHARPE JEE, YAN XT
1996a
Integrated platform for AI support of complex design (Part 1): Rapid development of schemes from first principles.
In SHARPE J (ed), A I system support for conceptual design. Proceedings of the 1995 Lancaster International Workshop on Engineering Design, Springer, p 170-188

BRACEWELL RH, CHAPLIN RV, LANGDON PM, LI M, OH VK, SHARPE JEE, YAN XT
1996b
Integrated platform for AI support of complex design (Part 2): Supporting the embodiment process.
In SHARPE J (ed), A I system support for conceptual design. Proceedings of the 1995 Lancaster International Workshop on Engineering Design, Springer, p 189-207

BRADLEY SR, AGOGINO AM
1991
Intelligent Real Time design methodology for catalog selection
American Society of Mechanical Engineers, Design Engineering Division (Publication) DE v 31 Sep 22-25 p 201-208

BRADY S, JUSTER N
1996
A computerised tool to create concept variants from function structures.
In SHARPE J (ed), A I system support for conceptual design. Proceedings of the 1995 Lancaster International Workshop on Engineering Design, Springer, p 208-226

BRITTON GA, McCALLION H
1989
Stafford Beer's viable system model: non-technical version
Appendix A, p214-244, Personal communication from H McCALLION, University of Canterbury, Christchurch, New Zealand

BROENINK JF
2001
Introduction to physical systems modelling with bond graphs
http://www.rt.el.utwente.nl/bnk/papers/BondGraphsV2.pdf accessed on 28 May 2001.

BROWN K
2001
What is Fuzzy Logic?
http://www.seanet.com/~ksbrown/kmath123.htm accessed at 11 Jan 2001

BS 5760
1996
Reliability of systems, equipment and components. Part 1: Dependability programme elements and tasks, BS 5760: Part 1: 1996 (EN 60300-2: 1996, IEC 300-2:1995)
British Standards Institution, London.

BS 7000: Part 1
1989
Design management systems. Guide to managing product design
British Standards Institution, London.

BURGESS JA
1987
Improving product reliability
Quality progress, Dec, p 47-54

CALANTONE RJ, DI BENEDETTO CA, SCHMIDT JB
1999
Using the Analytic Hierarchy Process in new product screening
Fournal of Product Innovation Management v 16 n 1 p 65-76

CALLAHAN TJ, MARLOWE DE
1993
Risk assessment and management as related to evaluation of medical devices. A view from the FDA
Journal of Long-Term Effects of Medical Implants v 3 n 4 p 269-276

CANDY L, EDMONDS E, PATRICK D
1996
Interactive knowledge support to conceptual design
In SHARPE J (ed), A I system support for conceptual design. Proceedings of the 1995 Lancaster International Workshop on Engineering Design, Springer, p 260-278

CANTU M
1998
Mastering Delpi 4
Sybex, San Francisco, ISBN 0-7821-2350-3

CARRASCOSA M, EPPINGRER S, WHITNEY D
1998
Using the Design Structure Matrix to Estimate Product Development Time.
Proceedings of the ASME Design Engineering Technical Conferences (Design Automation Conference), Atlanta, GA, Sept. 13-16
(http://web.mit.edu/dsm/publications_name.htm accessed at 21 Jan 2001)

CASTRO NC
1999
Test proceedure development for residential dishwashers
Appliance Engineer, Vol. 56, No. 8, p 70-76

CELLIER FE
2001
Controllab Products - Bond graphs
http://www.rt.el.utwente.nl/bnk/bondgraphs/bond.htm accessed on 29 May 2001.

CHITTISTER C, HAIMES YY
1993
Risk associated with software development: a holistic framework for assessment and management
IEEE Transactions on Systems, Man and Cybernetics v 23 n 3 May-Jun p 710-723

CLARKSON PJ, HAMILTON JR
2000
'Signposting': A parameter-driven task-based model of the design process.
Research in Engineering Design 2000(12): 18-38

CLEMEN RT
1996
Making hard decisions
2e, Duxbury Press

CLIBBON K, EDMONDS E, CANDY L
1996
Representing conceptual design knowledge with multi-layered logic
In SHARPE J (ed), A I system support for conceptual design. Proceedings of the 1995 Lancaster International Workshop on Engineering Design, Springer, p 93 - 108

COX S, TAIT N
1993
Reliability, Safety & Risk Management
Butterworth Heinemann, Guildford, ISBN 0 7506 1073 5

CRISP JDC
1986
Industrial Innovation and engineering education, parts I to III.
Acquiring Sensivity to Innovation, Mechanical Engineering Transactions of the IEAust., p 45 - 49

CROLL GJ
1995
Cost & schedule risk analysis in major engineering projects
IEE Colloquium (Digest) 183 Oct 30 IEE p 4/1-4/4 8

CROSSLAND R, WILLIAMS JHS, MCMAHON CA
1995
Object-oriented design model incorporating uncertainty for early risk assessment
ASME Database Symposium Sep 17-20 1995 p 93-101

CUNNINGHAM A, SMART R
1993
Computer-aided parts estimation
AI Magazine 14: 3, p 39-49

DAVIDSON J (ed)
1989
The Reliability of Mechanical Systems
Mechanical Engineering Publications, London

BORLAND DELPHI 5
1999
Developer's Guide, Borland Delphi 5 for Windows 98, Windows 95, & Windows NT
Inprise Corporation, 100 Enterprise Way, Scotts Valley, CA 95066-3249

DENG Y-M, BRITTON GA, TOR SB
1998
Design perspective of mechanical function and its object-oriented representation scheme
Engineering with Computers v 14 n 4 p 309-320

DENG Y-M, TOR SB, BRITTON GA
1999
Computerized design environment for functional modeling of mechanical products
Proceedings of the Symposium on Solid Modeling and Applications Jun 9-Jun 11 1999, p 1-12

DENG Y-M, TOR SB, BRITTON GA
2000
Abstracting and exploring functional design information for conceptual mechanical product design
Engineering with Computers 16 1 p 36-52

DEUTSCH MS
1982
Software verification and validation
Prentice Hall, New Jersey, ISBN 0-13-828072-7

DOMB E
1999
Problem solving using TRIZ
National Electronic Packaging and Production Conference-Proceedings of the Technical Program (West and East) v 2 Feb 21-Feb 25 1999 p 791-798

DUBOIS D, PRADE H
1999
Qualitative possibility theory and its applications to constraint satisfaction and decision under uncertainty
International Journal of Intelligent Systems v 14 n 1 p 45-61

DUCKWORTH B, CLARSON DS, VREEBURG J, ROSENTHAL L, POORTEMA K, HOOGSTEEN K
1998
Risk management
Water Supply v 16 n 1-2 p 467-471

D’AMBROSIO B, ULLMAN D
1995
Decision problem representations for collaborative design
Oregon State University, Corvallis, OR 97331 USA, downloaded from http://www.cs.orst.edu/~dambrosi/edss/info.html last updated: 19 April 1996 and accessed on 23 Jan 2001

EDER WE
1998
Design modeling - A Design Science approach (and why does industry not use it?)
Journal of Engineering Design, v 9 n 4 p 355-372

EIKENBERRY JA
1998
Linux & AI/AILIFE mini-howto
http://stommel.tamu.edu/~baum/linux/LDP/HOWTO/mini/AI-Alife.html last modified: 13 Jan 1998 and accessed on 21 Jan 2001

ENGHAGEN LK
1992
Fundamentals of Product Liability Law for Engineers
Industrial Press Inc. New York, ISBM 0-8311-3039-3

ESMIEU R, BUCQUET M
1995
Indicator for risk management
American Astronautical Society, Scientific Technology Series 93 Oct 2-6 p 153-159

FARAG WA, QUINTANA VH, LAMBERT-TORRES G.
1998
Genetic-based neuro-fuzzy approach for modeling and control of dynamical systems
IEEE Transactions on Neural Networks, v 9, n 5, Sep, p756-767

FINGER S, DIXON J
1989a
A review of research in mechanical engineering design. Part 1: Descriptive, prescriptive, and computer-based models of design process.
Research in Engineering Design, 1 p 51-67

FINGER S, DIXON J
1989b
A review of research in mechanical engineering design. Part 2: Representations, analysis, and design for the life cycle.
Research in Engineering Design, 1 p 121-137

FINGER S, FOX MS, PRINZ FB, RINDERLE JR
1992
Concurrent design
Applied Artificial Intelligence v 6 n 3 Jul-Sep p 257-283

FINLEY ED, FISHER DJ
1994
Project scheduling risk assessment using Monte Carlo methods
Cost Engineering v 36 n 10 p 26-28

FLETCHER M
1998
The mother of invention software
Eureka, April, p42-43

FROST RB
1999
Why does industry ignore design science?
Journal of Engineering Design, v10 n4 p297-304

FU Z, DE PENNINGTON A
1994
Geometric reasoning based on graph grammar parsing
Journal of Mechanical Design, Transactions Of the ASME v 116 n 3 Sept p 763-769

GARCIA AJ
1994
Fast track product development: the focus is on validation
Annual Technical Conference - ANTEC, Conference Proceedings n pt 3 p 2717-2718

GAUN X, MacCALLUM K
1996
Handling of positional information in a system for supporting early geometric design.
In SHARPE J (ed), A I system support for conceptual design. Proceedings of the 1995 Lancaster International Workshop on Engineering Design, Springer, p54 - 70

GIARRATANO J, RILEY G
1994
Expert systems: principles and programming
PWS Publishing Company, Boston

GIN B
2000
Personal communication, September 2000, Range & Dishwasher Division, Fisher & Paykel, Mosgiel, New Zealand.

GREENE LE
1989
Methodology for system warranty economic analysis
IEEE Proceedings of the National Aerospace and Electronics Conference v 3 May 22-26 1989 p 1210-1215

GUI J, MANTYLA M
1994
Functional understanding of assembly modelling
Computer Aided Design v 26 n 6 p 435-451

GUSTAFSSON A
1996
Customer focused product development by conjoint analysis and QFD
Linköping Studies in Science and Technology, Dissertation No. 418, Division of Quality Technology, Department of Mechanical Engineering, Linköping University, S-581 83 Linköping Sweden

GUYONNET D, COME B, PERROCHET P, PARRIAUX A
1999
Comparing two methods for addressing uncertainty in risk assessments
Journal of Environmental Engineering v 125 n 7 p 660-665

HAGUE M, TALEB-BENDIAB A, BRANDISH M
1996
An adaptive machine learning system for computer supported conceptual engineering design.
In SHARPE J (ed), A I system support for conceptual design. Proceedings of the 1995 Lancaster International Workshop on Engineering Design, Springer, p 1-16

HALES C
1994
Managing Engineering Design
Longman Scientific & Technical.

HALEY
1998
Answers to common questions about AI
The Haley Enterprise
http://www.haley.com/1266913081622528/ProductionSystems.html accessed at 23 Jan 2001

HANRATTY PJ, JOSEPH B, DUDUKOVIC MP
1992
Knowledge representation and reasoning in the presence of uncertainty in an expert system for laboratory reactor selection
Industrial & Engineering Chemistry Research v 31 n 1 p 228-238

GOURLAY JS
1984
Introduction to the formal treatment of testing
In HAUSEN H-L (Ed) Software validation, publ North-Holland, Amsterdam, ISBN 0-444-87593-X, p 67-72

HAYKIN S
1994
Neural Networks
Macmillan College Publishing Company, ISBN 0-02-352761-7

HEILMANN W-R
1990
Risk management and insurance
Forensic Engineering v 2 n 1-2 p 119-134

HENLEY EJ, KUMAMOTO H
1981
Reliability engineering and risk assessment.
Prentice Hall, USA.

HENRIKSEN ADP
1997
Technology assessment primer for management of technology
International Journal of Technology Management v 13 n 5-6 p 615-638

HERLING D, ULLMAN D, D'AMBROSIO B
1995
Engineering decision support system (EDSS)
ASME Design Engineering Division (Publication) DE v83 n2 pt1 p 619-626

HILL VL, BEALL CW, BLISCHKE WR
1991
Simulation model for warranty analysis
International Journal of Production Economics v 22 n 2 p 131-140

HUBKA V
1987
Principles of Engineering design
WDK1, Heurista, Zurich, ISBN 3-85693-018-3

HUBKA V, EDER WE
1988
Theory of Technical Systems
Springer-Verlag, ISBN 3-540-17451-6

HUBKA V, EDER WE
1996
Design Science
Springer-Verlag, Berlin, ISBN 3-540-19997-7.

HUDSON M, PARMEE I
1996
The application of genetic algorithms to conceptual design.
In SHARPE J (ed), A I system support for conceptual design. Proceedings of the 1995 Lancaster International Workshop on Engineering Design, Springer, p 17-36

HYATT LE, ROSENBERG LH
1988
Software metrics program for risk assessment
American Astronautical Society, Scientific Technology Series 95 p 117-133

IDEATION INTERNATIONAL
2001
http://www.ideationtriz.com/ accessed on 29 Jan 2001

IDEF
2000
IDEF
http://www2.umassd.edu/SWPI/IDEF/idefnote.html, accessed on 3 Nov 2000

IGNATIUS D
1999
What's amiss with Boeing?
The Press, Christchurch, New Zealand, Monday 8 Nov, p 13.

IGNIZIO J
1991
Introduction to expert systems
McGraw Hill

ISAACSON DN, REID S, BRENNAN JR
1991
Warranty cost-risk analysis
Proceedings of the Annual Reliability and Maintainability Symposium p 332-339

ISHII M, TOMIYAMA T
1996
A synthetic reasoning method based on a physical phenomenon knowledge base
In SHARPE J (ed), A I system support for conceptual design. Proceedings of the 1995 Lancaster International Workshop on Engineering Design, Springer, p 109-123

IVEZIC N, GARRETT JH
1998
Machine learning for simulation-based support of early collaborative design
Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM v 12 n 2 p 123-139

KANGAS, M
1996
Probabilistic risk assessment
Standardization News 24, 6 p 28-33

KANTROWITZ M, HORSTKOTTE E, JOSLYN C
1997
FAQ: Fuzzy Logic and Fuzzy Expert Systems 1/1
http://www.faqs.org/faqs/fuzzy-logic/part1/ Last-modified :14 Mar 1997 and accessed at 11 Jan 2001

KANTROWITZ M
1997
Expert System Shells,
Carnegie Mellon University,
http://www.cs.cmu.edu/Web/Groups/AI/html/faqs/ai/expert/part1/faq.html,
mkant+ai-faq@cs.cmu.edu

KEAT L, TAN C, MATHUR K
1996
Design model: towards an integrated representation for design semantics and syntax
In SHARPE J (ed), A I system support for conceptual design. Proceedings of the 1995 Lancaster International Workshop on Engineering Design, Springer, p 124-137

KEOLEIAN GA, GLANTSCHNIG WJ
1994
Life cycle design: AT&T demonstration project
IEEE International Symposium on Electronics & the Environment 1994 p 134-135

KENDALL J
1995
Using an industrial-design methodology to reduce the risks of medical product development
Medical Device and Diagnostic Industry 17,2 8pp

KERSTEN T
1996
“MODESSA”, a computer based conceptual design support system.
In SHARPE J (ed), A I system support for conceptual design. Proceedings of the 1995 Lancaster International Workshop on Engineering Design, Springer, p 241-259

KIMBLER DL
1997
IDEF0 Models
http://taylor.ces.clemson.edu/ie340/files/340-4.htm
Accessed on 3 Nov 2000

KINNISON RR
1985
Applied Extreme Value Statistics
Batelle Press (MacMillan), Columbus, ISBN 0 02 947630 5

KLEIJNEN JPC
1995
Verification and validation of simulation models
European Journal of Operational Research 82,1 p 145-162

KOSTETSKY O
1994
Facilitated, graphics & Monte Carlo based predictive project
IEEE International Engineering Management Conference Oct 17-19 p 177-184

KRISHNAN V, EPPINGER SD, WHITNEY DE
1995
Accelerating product development by the exchange of preliminary product design information
Journal of Mechanical Design, v 117 n 4 Dec p 491-498

KUIPERS B
1994
Qualitative reasoning
The MIT Press, Cambridge, ISBN 0-262-11190-X

LAD F
1996
Operational subjective statistical methods
John Wiley & Sons, New York, ISBN 0-471-14329-4

LAW AM, KELTON WD
1991
Simulation modelling and analysis
McGraw-Hill, ISBN 0-07-036698-5

MANN NR, FERTIG KW
1977
Efficient unbiased quantile estimators for mderate-size complete samples from extreme value and Weibull distributions: confidence bounds and tolerance prediction intervals.
Technometrics 19,1: p 87-93

MANNO I
1999
Introduction to the Monte Carlo method
Akademiai Kiado, Budapest, ISBN 963 05 7615 5

MAR BW
1991
System engineering and risk
Risk-Based Decision Making in Water Resources V, Proceedings of the Conference Nov 3-8 1991, p 304-310

MARSHALL HE
1990
Review of economic methods and risk analysis techniques for evaluating building investments
Batiment International/Building Research & Practice v 18 n 2 p 92-99

MARTINO JP
1994
Approach to balancing the risks of R&D performance goals
IEEE International Engineering Management Conference Oct 17-19 1994, p 164-166

MAZZUCHI TA, SOYER R
1992
Reliability assessment and prediction during product development
Proceedings of the Annual Reliability and Maintainability Symposium Mar 1992, 468-474

McCALLION H, BRITTON GA
1991
Effective management of 'Intellectual' teams with specific reference to design.
Journal of Engineering Design, 2:1:45-53

MEDLAND A
1996
Managing design and manufacturing constraints in a distributed industrial environment: The creation of a managed environment for engineering design.
In SHARPE J (ed), A I system support for conceptual design. Proceedings of the 1995 Lancaster International Workshop on Engineering Design, Springer, p336-348.

MILL HF
1994
Simplifying the implementation of QFD
IEE Colloquium (Digest) n 086 May 6 p 5/1-5/4

MIT
1999
The MIT Design Structure Matrix
Massachusetts Institute of Technology, http://web.mit.edu/dsm/
http://web.mit.edu/dsm/publications_name.htm

MONTAGUE DF
1990
Process risk evaluation. What method to use?
Reliability Engineering & System Safety v 29 n 1 1990 p 27-53

MOORE C, MILES J
1996
Integrated innovative computer systems for decision support in bridge design
In SHARPE J (ed), A I system support for conceptual design. Proceedings of the 1995 Lancaster International Workshop on Engineering Design, Springer: p227-240

NOGUCHI H
1998
Exploratory study on `Cross-Inference Method' for supporting idea generation process of industrial designers
International Conference on Knowledge-Based Intelligent Electronic Systems, p 384-389

O'CONNOR PDT
1981
Practical reliability engineering
John Wiley & Sons, ISBN 0 471 259 195

OH V, LANGDON P, SHARPE J
1994
Schemebuilder: an integrated computer environment for product design.
Lancaster Int. Workshop CACD 1994, Lancaster University, ISBN 0-901800 37 6, pp 339-362.

OH V, SHARPE J
1996
Conflict management in an interdisciplinary design environment.
In SHARPE J (ed), A I system support for conceptual design. Proceedings of the 1995 Lancaster International Workshop on Engineering Design, Springer: p298-318

OSSENBRUGGEN PJ
1994
Fundametal principles of systems analysis and decision making
John Wiley & Sons, New York, ISBN 0-471-52156-6

OSTERWEIL L
1984
Integrating the testing, analysis and debugging of programs
In HAUSEN H-L (Ed) Software validation, publ North-Holland, Amsterdam, ISBN 0-444-87593-X, p 73-102

OTTO KN, ANTONSSON EK
1993
Tuning parameters in engineering design
Journal of Mechanical Design, Transactions Of the ASME v 115 n 1 p 14-19

OWEN CL (ed)
1993
Design for integrity
insitute of Design Communications Center, Illinois Institute of Technology, Chicago, Illinois USA

PAHL G, BEITZ W
1988
Engineering Design: A systematic approach
Springer-Verlag (Berlin) and The Design Council (London). Wallace K (ed)

PEARCE R, COWLEY PH
1996
Use of fuzzy logic to describe constraints derived from engineering judgment in genetic algorithms
IEEE Transactions on Industrial Electronics, v43, n5, Oct, 1996, p 534-540

PELLISSIER JM, HAZEN GB, CHANG RW
1996
Continuous-risk decision analysis of total hip replacement
Journal of the Operational Research Society 47, 6 p 776-793

PEREGO A, RANGONE A
1996
On integrating tangible and intangible measures in AHP applications: a reference framework
Proceedings of the IEEE International Conference on Systems, Man and Cybernetics v 3 p 1836-1841

PETERSEN KE, LAURIDSEN K, RASMUSSEN B
1995
Use of functional modelling to assess the reliability of complex industrial systems in the design phase
American Society of Mechanical Engineers, Pressure Vessels and Piping Division (Publication) PVP v 320 p 155-161

PITTMAN RB
1996
Product & project planning: key to getting it right the first time
Wescon Conference Record Oct 22-24 1996 p 91-94

POWELL PB
1982
Software validation, verification and testing technique and tool reference guide.
NBS Special Publication 500-93

PRITSKER AAB
1990
Papers experiences perspectives
Systems publishing corporation, Indiana, USA, ISBN 0-938974-03-3

PUGH S
1991
Total Design, Integrated Methods for Successful Product Engineering.
Addison Wesley.

PUNCHES K
1996
Designing for reliability: a checklist
EDN, Nov 21, p149-156

QUELCH J, CAMERON IT
1994
Uncertainty representation and propagation in quantified risk assessment using fuzzy sets.
Journal of Loss Prevention in the Process Industries v 7 n 6 p 463-473

QUIN S, WIDERA GEO
1996
Uncertainty analysis in quantitative risk assessment
Journal of Pressure Vessel Technology, Transactions of the ASME 118, 1 p 121-124

RABINS M, ARDAYFIO d, FENVES S, SEIREG A, NADLER G, RICHARDSON H, CLARK Hl
1986
ASME Research, Design theory and methodology - a new discipline
Mechanical Engineering, August, p23-27

RAI SN, KREWSKI D
1998
Uncertainty and variability analysis in multiplicative risk models
Risk Analysis v 18 n 1 p 37-45

RAINE J
1998
The design process in industry - How good are our models and management practice?
Waves of Change conference, Gladstone Australia 28-30 Sept, p301-306

RAINE J, PONS D, WHYBREW K
2001
The design process and a methodology for system integrity
Proceedings of the Institution of Mechanical Engineers Vol 215 part B, p569-576.

RAINE J, WHYBREW K, DUNLOP G, VAN RIJ C, WARD D
1997
Management of undergraduate projects with specific reference to redesign of a Jacquard loom card-punch system
Journal of Engineering Design, v8 n4, p341-357

RAWLINSON G
1998
Driving innovation with a different way of thinking
Materials World v 6 n 4 Apr 1998 Inst of Materials p 207-209

RIDGMAN T
1996
Windows of opportunity: timing and entry strategies
Industrial Management and Data Systems v 96 n 5 p 26-31

ROSEN DW, DIXON JR, FINGER S
1992
Conversions of feature-based representations via graph grammar parsing.
American Society of Mechanical Engineers, Design Engineering Division (Publication) DE v 42 p 83-90

ROSEN D, DIXON J, FINGER S
1994
Conversions of feature-based design representations using graph grammar parsing.
Journal of Mechanical Design, Transactions Of the ASME v 116 n 3 p 785-792

ROSS SM
1997
Simulation
Second edition, Academic Press, San Diego, ISBN 0-12-598410-3

SANTILLAN-GUTIERREZ S, WRIGHT I
1996
Solution clustering with genetic algorithms and DFA: an experimental approach
In SHARPE J (ed), A I system support for conceptual design. Proceedings of the 1995 Lancaster International Workshop on Engineering Design, Springer: p37-53

SARPER H
1994
Capital rationing under risk: a chance constrained approach using uniformly distributed cash flows and available budgets.
Engineering Economist 39, 1 p 49-76

SCHMIDT LC, CAGAN J
1993
Recursive annealing: A computational model for machine design
Proceedings of the 5th International Conference on Design Theory and Methodology, American Society of Mechanical Engineers, Design Engineering Division (Publication) DE v 53 p 243-251

SCOTT MJ, ANTONSSON EK
1999
Arrow's Theorem and engineering design decision making
Research in Engineering Design - Theory, Applications, and Concurrent Engineering 11, 4 p 218-228

SHARPE J (ed)
1996
A I system support for conceptual design.
Proceedings of the 1995 Lancaster International Workshop on Engineering Design, Springer

SHU LH, WALLACE DR, FLOWERS WC
1996
Probabilistic methods in life-cycle design
IEEE International Symposium on Electronics & the Environment May 6-8 1996 p 7-12

SILVERMAN BG
1994
Unifying expert systems and the decision sciences
Operations Research v 42 n 3 p 393-413

SIU N
1990
Monte Carlo method for multiple parameter estimation in the presence of uncertain data
Reliability Engineering & System Safety v 28 n 1 p 59-98

SPRINGER MD
1979
The algebra of random variables
John Wiley & Sons, New York, ISBN 0-471-01406-0

STACHOWICZ M, BEALL L
2001
Fuzzy Logic Q&A
http://www.d.umn.edu/ece/lis/FzPack/fz_qa.html accessed at 11 Jan 2001

SUH N
1998
Axiomatic design theory for systems
Research in Engineering Design, 10: p 189-209

SYKES JB (ed)
1976
The Concise Oxford Dictionary of current English
Sixth Edition, Oxford University Press, London, ISBN 0-19-861121-8

SYSKI R
1989
Random Processes
Second edition, Marcel Dekker Inc, New York, ISBN 0-8247-8028-0

TANG M
1996
Development of an integrated AI system for conceptual design support.
In SHARPE J (ed), A I system support for conceptual design. Proceedings of the 1995 Lancaster International Workshop on Engineering Design, Springer: p153-169

TAYLOR BW
1999
Introduction to management science
Sixth edition, Prentice Hall, New Jersey USA, ISBN 0-13-918103-2

TECHOPTIMISER
1997
TechOptimiser 3.0 Software Manual, (approximate publication date as not explicit)
Invention Machine Corporation, 200 Portland Street, Boston Massachusetts 02114 USA, www.invention-machine.com

THORNTON AC, DONNELY S, ERTON B
2000
More than just robust design: why product development organizations still contend with variation and its impact on quality
Research in Engineering Design 2000(12): 127-143

UL 749 HOUSEHOLD DISHWASHERS
1997
Seventh edition, Underwriters Laboratory Inc., Northbrook, ISBN 0-7629-0197-7

ULLMAN D, D'AMBROSIO B
1995
A taxonomy for classifying engineering decision problems and support systems.
ASME Design Engineering Division (Publication) DE v83 n2 Pt1 p627-637

VOSE D
1996
Quantitative risk analysis
Wiley, ISBN 0-471-95803-4

WALLACE KM
1987
Studying the engineering design process in practice
International Congress on planning and design theory, Boston, USA 17-20 Aug, pp6

WALLEY P, MENEZES F, DE SOUZA C
1990
Uncertainty and indeterminacy in assessing the economic viability of energy options. A case study of solar heating systems in Brazil
Energy Systems and Policy v 14 n 4 1990 p 281-304

WANG Y, CHEN J
1995
Framework of decision analysis integrating qualitative reasoning and quantitative models
Proceedings of the IEEE International Conference on Systems, Man and Cybernetics 1 p 346-350

WEN YK, KANG YJ
1996
Design based on minimum expected life-cycle cost
Proceedings of the US-Japan Joint Seminar on Structural Optimization Apr 13 p 192-203

WHYBREW K
2001
Research overview
http://www.mech.canterbury.ac.nz/people/KWro.htm accessed on 17 Feb 2001

WOOD KL, ANTONSSON EK
1989
Computations with imprecise parameters in engineering design: background and theory
Journal of Mechanisms, Transmissions, and Automation in Design v 111 n 4 p 616-625

WOOD KL, ANTONSSON EK, BECK JL
1989
Comparing fuzzy and probability calculus for representing imprecision in preliminary engineering design
American Society of Mechanical Engineers, Design Engineering Division (Publication) DE v 17 Sep 17-21 1989 p99-105

WOLSTENHOLME EF
1999
Qualitative vs quantitative modelling: The evolving balance
Journal of the Operational Research Society v 50 n 4 p 422-428

YASSINE A, FALKENBURG D, CHELST K
1999
Engineering Design Management: An Information Structure Approach.
International Journal of Production Research, v37 n13 p 2957-2975

YASSINE AA, FALKENBURG DR
1999
A Framework for Design Process Specifications Management
Journal of Engineering Design, vol. 10, no. 3, p 223-234

ZHANG S, MCALLISTER J, DORRICOTT MG
1992
Probabilistic analysis of risk in mining projects - as easy as 123
Third Large Open Pit Mining Conference Aug 30-Sep 3 1992 p 455-459

ZHANG YF, FUH JYH
1998
Neural network approach for early cost estimation of packaging products
Computers & Industrial Engineering v 34 n 2 p 433-450

ZHONG G, DOONER M
1996
Use of visualisation and qualitative reasoning in configuring mechanical fasteners.
In SHARPE J (ed), A I system support for conceptual design. Proceedings of the 1995 Lancaster International Workshop on Engineering Design, Springer: p 362-376

ZLOTIN B, ZUSMAN A
1999
Managing Innovation Knowledge: The Ideation Approach to the Search, Development, and Utilization of Innovation Knowledge
http://www.ideationtriz.com/paper_Managing_Innovation.htm last updated 2 Jan 2001 and accessed on 29 Jan 2001

Index

7 PP Tools 73, 121
abstraction 139, 143
agent 127
AHP 30, 114
AI 130
algebra of random variables 167, 191
analogous solution 64
analysis uncertainty 212
analytic hierarchy process 62
Analytica 109, 220
analytical hierarchy process . . . 78, 95, 104, 218
anatomy 17
ancestor 278
anchor 300
anchoring 480
anticipate constraints 464
arcs 84
array 315
ART 53
artificial intelligence . . . 47, 125, 130, 139, 150, 154, 217, 337
assembly 92
assess 261
assessment 137, 142
assignment 105
AutoCAD 90
automotive 52
availability 300, 398, 480
Axiomatic design theory 40
back-propagation algorithm 60
backward chaining 49, 50
bath tub 477
Bayesian 217
Bayesian belief network 99
Bayesian probability 100, 110
belief 99, 120, 299
belief network 99
benchmarking 73, 116
bias 299, 480
blackboard 51, 69, 127
bond graph 84, 87, 88
bracket medians 301
break point 316
break-even 103
bundles of attributes 78
business environment 7
capital rationing 24, 166
case based reasoning 42, 51, 58
catalogue 42, 55, 84, 256
CBR 51, 58
central limit theorem 206, 269
certainty factor 62
class 85, 278
clusters 15
combinatorial 47, 220
completeness of knowledge 35
component 278
concept 5
concept checklist 74
concept design 5
concurrent 15
concurrent engineering 10, 72, 126
conditional knowledge 48
confidence 121, 231
configuration 39
configuration design 5
configuration tree 47
conflict 126
conjoint analysis 74, 78, 95, 218
consequence diagram 446
conservation of energy 88
constraint 4, 462
constraint relaxation 127
continuous simulation 81
contradiction 64
convolution of densities 168
correlate 231
correlation 241, 246, 478
cost 399
coupled tasks 16
critical path 15
cumulative distribution 190
customer 4, 73
database 43, 54, 127
debug 315
decision 32
decision analysis 2, 98, 218, 239, 299
decision table 221, 226, 340
decision theory 155, 165
decision tree 47, 49, 100
decomposition 2, 39, 68, 79, 138
decoupling 108
Delphi 226, 276, 314
demand 115
Dempster-Shafer belief function 121
dependability 398
dependency 16
design 4
Design for System Integrity 156
Design Integrity 156
design intent . . 31, 91, 122, 132, 136, 334, 464
design of experiments 79, 95
design process 4
Design Science 17
design structure matrix 14, 218
design team 121
designed experiment 79, 95
detailed design 5

determinism 35, 139
differential equations 111
direct user benefit 115
directed graph 59
discount rate 114
discrete event simulation 81
discursive 4, 44
distributed design . . . 10, 39, 65, 68, 126, 138, 139
drill down 273
DSI 156
DSM 14
dynamic array 315
dynamic network model 100
dynamic system 111
early design 97
economic feasibility 112
edges 84
electric shock 432
embodiment design 5
energy usage 334
engineering design 4
entropy 476
estimation 52
expert system . 50, 97, 128, 150, 154, 217, 330, 337
exponential 167
Extended Pearson-Tukey 300
failure mode and effect analysis 2, 401
failure rate 166
fault tree 129
fault tree analysis 2, 401
feasible region 103
feature 91, 92
feedback analysis 108
fire 434
first order predicate logic 49
fitness 142
fitness factor 55
flowdown 2
FMEA 2, 401
FMECA 97, 131
focus groups 73
forward chaining 50
frame 52
FTA 2, 401
full profile approach 78
function 4, 138
functional decomposition 39, 45, 87
functional model 252
functional modelling 40, 81, 131, 137, 150, 154, 158, 164, 216
fuzzy arithmetic 172
fuzzy logic 96
fuzzy set 62, 111, 172
fuzzy theory . . . 56, 79, 96, 172, 217, 218, 223
Gantt 12
generic design activity 27
genetic algorithm 55, 57, 158, 322
geometric 91
geometric reasoning 91
geometry 92
goal programming 104
grammar 54, 92, 133, 138
graph 65, 84, 92
graph theory 87, 137
group interview 73
hazard 422
HAZOP 129, 401
heuristic 127
histogram 189
House of Quality 76
Hugin 101, 219
ICAD 133
ideality 63
ignorant designer 150
ill-structured design 68
image map 273
imprecision 79
incompleteness of knowledge 147, 234
independence 225
independent 10
inference engine 50
influence diagram 99, 109, 198
information abstraction 35, 147
inheritance 52
injury 437
instance 54, 85
integer programming 104
integration definition 13
integrity 151, 152, 252, 311
internal rate of return 114
interval 111, 172
interval arithmetic 206
intuitive 44
inventive principles 64
IO 296
joint density 167
Kano model 73
key characteristic 2, 22, 250, 257
knowledge 121
knowledge engineering 48
knowledge source 51
Laplace 260
Latin Hypercube Sampling 109
layout design 5
LCA 131
liability 445
library 85
life cycle 4, 131, 136, 146, 150, 399
life cycle assessment 131
life cycle cost 320
linear algebra 15
linear model 6
linear programming 103
Lisp 53, 92

load-capability interference 166
location 92
location parameter 477, 502
logic based languages 49
loss function 96
manage 250, 261, 399
management 10, 130
management science 103
map 221, 226
Maple 172
mapping 41, 42
market analysis 78
marketing 73
Markov 16, 167
Markov analysis 107
MathCad 172
maximin 260
mean 190
median 190
memory leak 316
Method of Imprecision 97
mode 87, 189
Monte Carlo . . . 81, 95, 97, 108, 131, 155, 173, 216, 239, 322, 389
morphological analysis 46, 67, 87
motivational bias 300
motor vehicle 42
multi-criteria decision 113, 116
multi-criteria decision making 104, 127
multi-parameter sensitivity analysis 321
multiple viewpoints 477
natural language 85
negation 113
net present value 115, 166, 320
net worth 103, 115
network 99, 105
neural network 42, 56, 59
node 84
non-linear programming 105
normal distribution 495
novice 150
NPV 320
object oriented 59, 85
object oriented programming . . . 48, 52, 54, 276
obsolescence 250
operating condition 45
operations research 103
optimisation 107
organ 17
organisation process 7
overconfidence 300
overlapping 16
paired comparisons 78
parametric design 6
parse 54, 92, 484
participant observer 11
particle 338
partitioning algorithms 15
Pascal 276
patch 338
perceptron 59
performance 334, 399
persistent question 113
PERT 13, 391, 476
phase diagram 12
plausibility 121
point sort 487
positioning analysis 73
present value 114, 320, 323
prevision 212
probabilistic computation 130, 173
probabilistic reasoning 16, 99, 130
probability calculus 167
probability density 189
probability distribution 108
probe 313
process variability . 2, 35, 87, 96, 145, 147, 153, 212, 213, 276
product 73
production 49
production system 50
professional judgement 31
project management 12
Prolog 42, 49
propagation of change 87, 470
proportional sort 487
Pugh 114, 123
QFD 30, 74, 114, 159, 218, 223
QRA 129, 157, 165
qualitative 129, 136, 147, 234
qualitative differential equation 111
qualitative evaluation 86
qualitative physics 83
qualitative possibility 101
qualitative probability 17
qualitative risk analysis 129
qualitative simulation 111
quality 73, 80
quality function deployment 74, 217
quantitative 108, 111, 122, 129, 138
quantitative methods 103
quantitative risk assessment 129, 157, 165
questionnaire survey 73
queueing theory 107
random variable 130, 167
rationale 122
recursive annealing 56
reinforcement learning 60
relational database 43, 90
reliability 2, 97, 131, 141, 145, 166, 398
representativeness 299
resolution 486
Rete algorithm 50
risk 11, 16, 80, 107, 250
risk analysis 129
risk assessment 2, 129

risk averse 259
risk phobic 259
robust 104, 152
rule 50, 52
safety 112, 132, 422
scheme 88
Schemebuilder 88, 133
semantics 138
semi-quantitative simulation 239
sensitivity 16
sensitivity analysis 16, 94, 164, 321, 388
seven product planning tools 73
shape grammar 54
shells 51
signposting 17, 257
simplex method 103
simulated annealing 56
simulation 158
Simulink 89
simultaneous engineering 10
social choice 128
Software integrity 311
solution generation 150, 464
specification 312
state 87
stereotyping 299
stochastic uncertainty 145
structure design 5
subjective probability 299
supervised learning 60
supply 115
support 121, 148
symbolic execution 313
system modelling 150
systematic doubting 113
systematic variation 47
systems analysis 107, 115
table-type conceptualising 74
Taguchi 96
team 16
tear 16
technical feasibility 112
technical process 17
technical system 17, 66, 82
testing 311
thread 466
time 399
TIPS 30, 63
tornado diagram 94, 164, 321
trade-off 65, 79, 114
transportation 105
trimmed 65
TRIZ 30, 63
truth table 101
tuning parameters 97
Turing Test 312
uncertainty 66, 81, 144, 150, 212
uncertainty of analysis 2, 35, 87, 144, 153, 212, 214, 276, 470
underestimating 300
unresolved 224
utility 4, 99, 121, 123
validation 54, 311, 312
value analysis 77
variability 16
variability of process 145
verification 311
vertices 84
viewpoint 146, 147, 251, 334, 477
voice of the customer 4, 73
warranty exposure 414
wash performance 334, 338, 358
washing index 335
water consumption 334
weak order 115, 121, 124
weight . 43, 70, 77, 78, 104, 114, 127, 142, 153
what-if analysis 164

Author and Supervisors

Dirk Pons

Pr Eng (ECSA), Reg BioMed Eng (SAMDC), MIPENZ, MSc Medicine (Biomedical Engineering) (CapeTown), BSc Eng (Mech) (Natal)

Professional interests: Machine and product design, including consumer products and special purpose machine design, reliability.

Professor John Raine

BE(Hons), PhD, CEng, FIMech.E, FIPENZ, MSAE-A.

Professional Interests: Machine design and design methodology; alternative energy systems; engines and vehicles; dynamometer design and performance modelling; vehicle crash reconstruction.

Associate Professor Ken Whybrew

BSc(Eng)(Hons), PhD(Nott), CEng, MIMech.E.

Professional Interests: Computer NC systems; flexible manufacturing systems; computer-aided-engineering.

"*My research is directed at organisational techniques and software tools to shorten the time taken by New Zealand manufacturers to introduce new products to international markets.*"